libstdc++

Generated by Doxygen 1.13.2

1 Deprecated List 1

2 Todo List 2

3 Topic Documentation 4

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

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

3.15.5 Memory . . . 317
3.15.6 Metaprogramming . . . 346
3.15.7 Rational Arithmetic . . . 354
3.15.8 Time . . . 356

4 Namespace Documentation **377**

4.1 __gnu_cxx Namespace Reference . . . 377
4.1.1 Detailed Description . . . 394
4.1.2 Typedef Documentation . . . 394
4.1.3 Function Documentation . . . 394
4.2 __gnu_cxx::__detail Namespace Reference . . . 404
4.2.1 Detailed Description . . . 404
4.2.2 Function Documentation . . . 404
4.3 __gnu_cxx::typelist Namespace Reference . . . 405
4.3.1 Detailed Description . . . 405
4.3.2 Function Documentation . . . 405
4.4 __gnu_debug Namespace Reference . . . 405
4.4.1 Detailed Description . . . 411
4.4.2 Typedef Documentation . . . 411
4.4.3 Enumeration Type Documentation . . . 412
4.4.4 Function Documentation . . . 412
4.5 __gnu_internal Namespace Reference . . . 414
4.5.1 Detailed Description . . . 414
4.6 __gnu_parallel Namespace Reference . . . 414
4.6.1 Detailed Description . . . 422
4.6.2 Typedef Documentation . . . 422
4.6.3 Enumeration Type Documentation . . . 423
4.6.4 Function Documentation . . . 423
4.6.5 Variable Documentation . . . 462
4.7 __gnu_pbds Namespace Reference . . . 462
4.7.1 Detailed Description . . . 464
4.8 __gnu_sequential Namespace Reference . . . 464
4.8.1 Detailed Description . . . 464
4.9 abi Namespace Reference . . . 464
4.9.1 Detailed Description . . . 464
4.10 std Namespace Reference . . . 464
4.10.1 Detailed Description . . . 578
4.10.2 Typedef Documentation . . . 578
4.10.3 Enumeration Type Documentation . . . 580

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

4.24 std::tr1 Namespace Reference . . . 721
4.24.1 Detailed Description . . . 724
4.24.2 Function Documentation . . . 724
4.25 std::tr1::__detail Namespace Reference . . . 727
4.25.1 Detailed Description . . . 727
4.26 std::tr2 Namespace Reference . . . 727
4.26.1 Detailed Description . . . 728
4.27 std::tr2::__detail Namespace Reference . . . 728
4.27.1 Detailed Description . . . 728

5 Class Documentation **728**

5.1 __gnu_parallel::__accumulate_binop_reduct< _BinOp > Struct Template Reference . . . 728
5.1.1 Detailed Description . . . 729
5.2 __gnu_parallel::__accumulate_selector< _It > Struct Template Reference . . . 729
5.2.1 Detailed Description . . . 729
5.2.2 Member Function Documentation . . . 730
5.2.3 Member Data Documentation . . . 730
5.3 __gnu_parallel::__adjacent_difference_selector< _It > Struct Template Reference . . . 730
5.3.1 Detailed Description . . . 731
5.3.2 Member Data Documentation . . . 731
5.4 __gnu_parallel::__adjacent_find_selector Struct Reference . . . 731
5.4.1 Detailed Description . . . 732
5.4.2 Member Function Documentation . . . 732
5.5 __gnu_cxx::__alloc_traits< _Alloc, typename > Struct Template Reference . . . 732
5.5.1 Detailed Description . . . 734
5.5.2 Member Typedef Documentation . . . 734
5.5.3 Member Function Documentation . . . 735
5.6 std::__basic_future< _Res > Class Template Reference . . . 738
5.6.1 Detailed Description . . . 739
5.6.2 Member Function Documentation . . . 740
5.7 __gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType > Class Template Reference . . . 740
5.7.1 Detailed Description . . . 741
5.7.2 Member Typedef Documentation . . . 741
5.8 __gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType > Class Template Reference . . . 741
5.8.1 Detailed Description . . . 742
5.8.2 Member Typedef Documentation . . . 742
5.9 std::__codecvt_abstract_base< _InternT, _ExternT, _StateT > Class Template Reference . . . 742
5.9.1 Detailed Description . . . 743

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

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

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

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

5.65 std::_Bind_result< _Result, _Signature > Class Template Reference . . . 863
5.65.1 Detailed Description . . . 863
5.66 __gnu_cxx::__detail::_Bitmap_counter< _Tp > Class Template Reference . . . 863
5.66.1 Detailed Description . . . 864
5.67 std::__detail::_BracketMatcher< _TraitsT, __icase, __collate > Struct Template Reference . . . 864
5.67.1 Detailed Description . . . 864
5.68 __gnu_cxx::_Caster< _ToType > Struct Template Reference . . . 865
5.68.1 Detailed Description . . . 865
5.69 __gnu_cxx::_Char_types< _CharT > Struct Template Reference . . . 865
5.69.1 Detailed Description . . . 865
5.70 __gnu_pbds::detail::pat_trie_base::_CIter< Node, Leaf, Head, Inode, Is_Forward_Iterator > Class Template Reference . . . 865
5.70.1 Detailed Description . . . 867
5.71 std::__detail::_Compiler< _TraitsT > Class Template Reference . . . 867
5.71.1 Detailed Description . . . 868
5.72 std::__parallel::_CRandNumber< _MustBeInt > Struct Template Reference . . . 868
5.72.1 Detailed Description . . . 868
5.73 std::_Deque_base< _Tp, _Alloc > Class Template Reference . . . 868
5.73.1 Detailed Description . . . 869
5.73.2 Member Function Documentation . . . 869
5.74 std::_Deque_iterator< _Tp, _Ref, _Ptr > Struct Template Reference . . . 870
5.74.1 Detailed Description . . . 871
5.74.2 Member Function Documentation . . . 871
5.75 __gnu_parallel::_DRandomShufflingGlobalData< _RAIter > Struct Template Reference . . . 871
5.75.1 Detailed Description . . . 872
5.75.2 Constructor & Destructor Documentation . . . 872
5.75.3 Member Data Documentation . . . 872
5.76 __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator > Struct Template Reference . . . 873
5.76.1 Detailed Description . . . 873
5.76.2 Member Data Documentation . . . 873
5.77 __gnu_parallel::_DummyReduct Struct Reference . . . 874
5.77.1 Detailed Description . . . 874
5.78 __gnu_debug::_Equal_to< _Type > Class Template Reference . . . 874
5.78.1 Detailed Description . . . 875
5.79 __gnu_parallel::_EqualFromLess< _T1, _T2, _Compare > Class Template Reference . . . 875
5.79.1 Detailed Description . . . 875
5.79.2 Member Typedef Documentation . . . 876
5.80 __gnu_parallel::_EqualTo< _T1, _T2 > Struct Template Reference . . . 876
5.80.1 Detailed Description . . . 877

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

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

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

5.126.1 Detailed Description . . . 931
5.126.2 Member Typedef Documentation . . . 931
5.127 std::__fwdlist::_Node< _ValPtr > Struct Template Reference . . . 931
5.127.1 Detailed Description . . . 932
5.128 __gnu_pbds::detail::pat_trie_base::_Node_base< _ATraits, Metadata > Struct Template Reference . . . 933
5.128.1 Detailed Description . . . 933
5.129 std::__fwdlist::_Node_base< _VoidPtr > Struct Template Reference . . . 934
5.129.1 Detailed Description . . . 934
5.130 __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > Class Template Reference . . . 934
5.130.1 Detailed Description . . . 935
5.130.2 Member Typedef Documentation . . . 935
5.130.3 Member Function Documentation . . . 936
5.131 __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > Class Template Reference . . . 937
5.131.1 Detailed Description . . . 938
5.131.2 Member Typedef Documentation . . . 938
5.131.3 Member Function Documentation . . . 938
5.132 __gnu_debug::_Not_equal_to< _Type > Class Template Reference . . . 940
5.132.1 Detailed Description . . . 940
5.133 std::_Not_fn< _Fn > Class Template Reference . . . 940
5.133.1 Detailed Description . . . 941
5.134 __gnu_parallel::_Nothing Struct Reference . . . 941
5.134.1 Detailed Description . . . 941
5.134.2 Member Function Documentation . . . 941
5.135 __gnu_parallel::_Piece< _DifferenceTp > Struct Template Reference . . . 941
5.135.1 Detailed Description . . . 942
5.135.2 Member Data Documentation . . . 942
5.136 std::_Placeholder< _Num > Struct Template Reference . . . 942
5.136.1 Detailed Description . . . 942
5.137 __gnu_parallel::_Plus< _Tp1, _Tp2, _Result > Struct Template Reference . . . 942
5.137.1 Detailed Description . . . 943
5.137.2 Member Typedef Documentation . . . 943
5.138 __gnu_parallel::_PMWMSSortingData< _RAIter > Struct Template Reference . . . 944
5.138.1 Detailed Description . . . 944
5.138.2 Member Data Documentation . . . 944
5.139 __gnu_cxx::_Pointer_adapter< _Storage_policy > Class Template Reference . . . 945
5.139.1 Detailed Description . . . 947
5.140 __gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp > Class Template Reference . . . 947
5.140.1 Detailed Description . . . 948

5.140.2 Constructor & Destructor Documentation . . . 948
5.140.3 Member Function Documentation . . . 948
5.141 __gnu_parallel::_PseudoSequenceIterator< _Tp, _DifferenceTp > Class Template Reference . . . 948
5.141.1 Detailed Description . . . 949
5.142 __gnu_parallel::_QSBThreadLocal< _RAIter > Struct Template Reference . . . 949
5.142.1 Detailed Description . . . 949
5.142.2 Member Typedef Documentation . . . 950
5.142.3 Constructor & Destructor Documentation . . . 950
5.142.4 Member Data Documentation . . . 950
5.143 std::__detail::_Quoted_string< _String, _CharT > Struct Template Reference . . . 951
5.143.1 Detailed Description . . . 951
5.144 __gnu_parallel::_RandomNumber Class Reference . . . 951
5.144.1 Detailed Description . . . 951
5.144.2 Constructor & Destructor Documentation . . . 951
5.144.3 Member Function Documentation . . . 952
5.145 __gnu_cxx::_Relative_pointer_impl< _Tp > Class Template Reference . . . 952
5.145.1 Detailed Description . . . 953
5.146 __gnu_cxx::_Relative_pointer_impl< const _Tp > Class Template Reference . . . 954
5.146.1 Detailed Description . . . 954
5.147 __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp > Class Template Reference . . . 954
5.147.1 Detailed Description . . . 955
5.147.2 Constructor & Destructor Documentation . . . 955
5.147.3 Member Function Documentation . . . 955
5.148 __gnu_debug::_Safe_container< _SafeContainer, _Alloc, _SafeBase, _IsCxx11AllocatorAware > Class Template Reference . . . 956
5.148.1 Detailed Description . . . 958
5.149 __gnu_debug::_Safe_forward_list< _SafeSequence > Class Template Reference . . . 958
5.149.1 Detailed Description . . . 958
5.149.2 Member Function Documentation . . . 959
5.149.3 Member Data Documentation . . . 959
5.150 __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category > Class Template Reference . . . 960
5.150.1 Detailed Description . . . 962
5.150.2 Constructor & Destructor Documentation . . . 962
5.150.3 Member Function Documentation . . . 963
5.150.4 Member Data Documentation . . . 968
5.151 __gnu_debug::_Safe_iterator_base Class Reference . . . 969
5.151.1 Detailed Description . . . 971
5.151.2 Constructor & Destructor Documentation . . . 971
5.151.3 Member Function Documentation . . . 972

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

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

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

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

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

5.218.5 Member Function Documentation . . . 1120
5.218.6 Member Data Documentation . . . 1155
5.219 __gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc > Class Template Reference . . . 1160
5.219.1 Detailed Description . . . 1161
5.220 __gnu_pbds::basic_hash_tag Struct Reference . . . 1161
5.220.1 Detailed Description . . . 1162
5.221 std::basic_ifstream< _CharT, _Traits > Class Template Reference . . . 1162
5.221.1 Detailed Description . . . 1168
5.221.2 Member Typedef Documentation . . . 1168
5.221.3 Member Enumeration Documentation . . . 1169
5.221.4 Constructor & Destructor Documentation . . . 1169
5.221.5 Member Function Documentation . . . 1171
5.221.6 Member Data Documentation . . . 1197
5.222 __gnu_pbds::basic_invalidation_guarantee Struct Reference . . . 1202
5.222.1 Detailed Description . . . 1202
5.223 std::basic_ios< _CharT, _Traits > Class Template Reference . . . 1203
5.223.1 Detailed Description . . . 1206
5.223.2 Member Typedef Documentation . . . 1207
5.223.3 Member Enumeration Documentation . . . 1209
5.223.4 Constructor & Destructor Documentation . . . 1209
5.223.5 Member Function Documentation . . . 1209
5.223.6 Member Data Documentation . . . 1220
5.224 std::basic_iostream< _CharT, _Traits > Class Template Reference . . . 1225
5.224.1 Detailed Description . . . 1230
5.224.2 Member Typedef Documentation . . . 1231
5.224.3 Member Enumeration Documentation . . . 1232
5.224.4 Constructor & Destructor Documentation . . . 1232
5.224.5 Member Function Documentation . . . 1232
5.224.6 Member Data Documentation . . . 1267
5.225 std::basic_istream< _CharT, _Traits > Class Template Reference . . . 1271
5.225.1 Detailed Description . . . 1276
5.225.2 Member Typedef Documentation . . . 1277
5.225.3 Member Enumeration Documentation . . . 1278
5.225.4 Constructor & Destructor Documentation . . . 1278
5.225.5 Member Function Documentation . . . 1278
5.225.6 Member Data Documentation . . . 1304
5.226 std::basic_istringstream< _CharT, _Traits, _Alloc > Class Template Reference . . . 1309
5.226.1 Detailed Description . . . 1314

5.226.2 Member Typedef Documentation . . . 1314
5.226.3 Member Enumeration Documentation . . . 1317
5.226.4 Constructor & Destructor Documentation . . . 1317
5.226.5 Member Function Documentation . . . 1318
5.226.6 Member Data Documentation . . . 1343
5.227 std::basic_ofstream< _CharT, _Traits > Class Template Reference . . . 1348
5.227.1 Detailed Description . . . 1353
5.227.2 Member Typedef Documentation . . . 1353
5.227.3 Member Enumeration Documentation . . . 1354
5.227.4 Constructor & Destructor Documentation . . . 1354
5.227.5 Member Function Documentation . . . 1356
5.227.6 Member Data Documentation . . . 1378
5.228 std::basic_ostream< _CharT, _Traits > Class Template Reference . . . 1382
5.228.1 Detailed Description . . . 1387
5.228.2 Member Typedef Documentation . . . 1387
5.228.3 Member Enumeration Documentation . . . 1389
5.228.4 Constructor & Destructor Documentation . . . 1389
5.228.5 Member Function Documentation . . . 1389
5.228.6 Member Data Documentation . . . 1408
5.229 std::basic_ostringstream< _CharT, _Traits, _Alloc > Class Template Reference . . . 1412
5.229.1 Detailed Description . . . 1418
5.229.2 Member Typedef Documentation . . . 1419
5.229.3 Member Enumeration Documentation . . . 1420
5.229.4 Constructor & Destructor Documentation . . . 1420
5.229.5 Member Function Documentation . . . 1421
5.229.6 Member Data Documentation . . . 1442
5.230 std::basic_regex< _Ch_type, _Rx_traits > Class Template Reference . . . 1446
5.230.1 Detailed Description . . . 1447
5.230.2 Constructor & Destructor Documentation . . . 1448
5.230.3 Member Function Documentation . . . 1451
5.231 std::basic_streambuf< _CharT, _Traits > Class Template Reference . . . 1455
5.231.1 Detailed Description . . . 1458
5.231.2 Member Typedef Documentation . . . 1459
5.231.3 Constructor & Destructor Documentation . . . 1460
5.231.4 Member Function Documentation . . . 1460
5.231.5 Member Data Documentation . . . 1472
5.232 __gnu_debug::basic_string< _CharT, _Traits, _Allocator > Class Template Reference . . . 1473
5.232.1 Detailed Description . . . 1480
5.232.2 Member Function Documentation . . . 1481

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

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

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

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

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

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

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

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

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

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

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

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

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

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

5.415 std::error_code Class Reference . . . 2111
5.415.1 Detailed Description . . . 2112
5.415.2 Constructor & Destructor Documentation . . . 2112
5.415.3 Member Function Documentation . . . 2112
5.416 std::error_condition Class Reference . . . 2113
5.416.1 Detailed Description . . . 2114
5.416.2 Constructor & Destructor Documentation . . . 2114
5.416.3 Member Function Documentation . . . 2114
5.417 __gnu_parallel::exact_tag Struct Reference . . . 2115
5.417.1 Detailed Description . . . 2115
5.417.2 Member Function Documentation . . . 2115
5.418 std::exception Class Reference . . . 2116
5.418.1 Detailed Description . . . 2117
5.418.2 Member Function Documentation . . . 2117
5.419 std::__unspecified__::exception_ptr Class Reference . . . 2117
5.419.1 Detailed Description . . . 2118
5.420 std::exception_ptr Class Reference . . . 2118
5.420.1 Detailed Description . . . 2118
5.421 std::exponential_distribution< _RealType > Class Template Reference . . . 2118
5.421.1 Detailed Description . . . 2119
5.421.2 Member Typedef Documentation . . . 2120
5.421.3 Constructor & Destructor Documentation . . . 2120
5.421.4 Member Function Documentation . . . 2120
5.421.5 Friends And Related Symbol Documentation . . . 2121
5.422 std::extent< typename, _Uint > Struct Template Reference . . . 2121
5.422.1 Detailed Description . . . 2122
5.423 std::extreme_value_distribution< _RealType > Class Template Reference . . . 2122
5.423.1 Detailed Description . . . 2123
5.423.2 Member Typedef Documentation . . . 2123
5.423.3 Member Function Documentation . . . 2123
5.423.4 Friends And Related Symbol Documentation . . . 2125
5.424 std::locale::facet Class Reference . . . 2125
5.424.1 Detailed Description . . . 2127
5.424.2 Constructor & Destructor Documentation . . . 2127
5.425 std::filesystem::file_status Class Reference . . . 2128
5.425.1 Detailed Description . . . 2128
5.426 std::experimental::filesystem::v1::filesystem_error Class Reference . . . 2128
5.426.1 Detailed Description . . . 2129
5.426.2 Member Function Documentation . . . 2129

5.427 std::filesystem::filesystem_error Class Reference . . . 2130
5.427.1 Detailed Description . . . 2130
5.427.2 Member Function Documentation . . . 2130
5.428 __gnu_parallel::find_tag Struct Reference . . . 2131
5.428.1 Detailed Description . . . 2131
5.429 std::fisher_f_distribution< _RealType > Class Template Reference . . . 2131
5.429.1 Detailed Description . . . 2132
5.429.2 Member Typedef Documentation . . . 2132
5.429.3 Member Function Documentation . . . 2133
5.429.4 Friends And Related Symbol Documentation . . . 2133
5.430 __gnu_cxx::forced_error Struct Reference . . . 2134
5.430.1 Detailed Description . . . 2135
5.430.2 Member Function Documentation . . . 2135
5.431 std::forward_iterator_tag Struct Reference . . . 2135
5.431.1 Detailed Description . . . 2136
5.432 std::__debug::forward_list< _Tp, _Alloc > Class Template Reference . . . 2136
5.432.1 Detailed Description . . . 2139
5.433 std::forward_list< _Tp, _Alloc > Class Template Reference . . . 2139
5.433.1 Detailed Description . . . 2141
5.433.2 Constructor & Destructor Documentation . . . 2142
5.433.3 Member Function Documentation . . . 2145
5.434 std::fpos< _StateT > Class Template Reference . . . 2157
5.434.1 Detailed Description . . . 2158
5.434.2 Constructor & Destructor Documentation . . . 2158
5.434.3 Member Function Documentation . . . 2158
5.435 __gnu_cxx::free_list Class Reference . . . 2159
5.435.1 Detailed Description . . . 2160
5.435.2 Member Function Documentation . . . 2160
5.436 std::from_chars_result Struct Reference . . . 2161
5.436.1 Detailed Description . . . 2161
5.437 std::front_insert_iterator< _Container > Class Template Reference . . . 2161
5.437.1 Detailed Description . . . 2162
5.437.2 Member Typedef Documentation . . . 2162
5.437.3 Constructor & Destructor Documentation . . . 2163
5.437.4 Member Function Documentation . . . 2163
5.438 std::function< _Res(_ArgTypes...)> Class Template Reference . . . 2164
5.438.1 Detailed Description . . . 2165
5.438.2 Constructor & Destructor Documentation . . . 2165
5.438.3 Member Function Documentation . . . 2166

5.439 std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> Class Template Reference . . . 2170
5.439.1 Detailed Description . . . 2170
5.439.2 Constructor & Destructor Documentation . . . 2171
5.439.3 Member Function Documentation . . . 2172
5.440 std::future< _Res > Class Template Reference . . . 2172
5.440.1 Detailed Description . . . 2173
5.440.2 Constructor & Destructor Documentation . . . 2173
5.440.3 Member Function Documentation . . . 2173
5.441 std::future< _Res & > Class Template Reference . . . 2173
5.441.1 Detailed Description . . . 2175
5.441.2 Constructor & Destructor Documentation . . . 2175
5.441.3 Member Function Documentation . . . 2175
5.442 std::future< void > Class Reference . . . 2175
5.442.1 Detailed Description . . . 2177
5.442.2 Constructor & Destructor Documentation . . . 2177
5.442.3 Member Function Documentation . . . 2177
5.443 std::future_error Class Reference . . . 2177
5.443.1 Detailed Description . . . 2178
5.443.2 Member Function Documentation . . . 2178
5.444 std::gamma_distribution< _RealType > Class Template Reference . . . 2178
5.444.1 Detailed Description . . . 2179
5.444.2 Member Typedef Documentation . . . 2180
5.444.3 Constructor & Destructor Documentation . . . 2180
5.444.4 Member Function Documentation . . . 2180
5.444.5 Friends And Related Symbol Documentation . . . 2181
5.445 std::geometric_distribution< _IntType > Class Template Reference . . . 2182
5.445.1 Detailed Description . . . 2183
5.445.2 Member Typedef Documentation . . . 2183
5.445.3 Member Function Documentation . . . 2183
5.445.4 Friends And Related Symbol Documentation . . . 2184
5.446 __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc > Class Template Reference . . . 2185
5.446.1 Detailed Description . . . 2186
5.446.2 Constructor & Destructor Documentation . . . 2186
5.447 __gnu_pbds::gp_hash_tag Struct Reference . . . 2190
5.447.1 Detailed Description . . . 2191
5.448 __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy > Class Template Reference . . . 2191
5.448.1 Detailed Description . . . 2194

5.448.2 Member Enumeration Documentation . . . 2194
5.448.3 Member Function Documentation . . . 2194
5.449 std::chrono::gps_clock Class Reference . . . 2196
5.449.1 Detailed Description . . . 2196
5.450 std::greater< _Tp > Struct Template Reference . . . 2197
5.450.1 Detailed Description . . . 2197
5.450.2 Member Typedef Documentation . . . 2197
5.451 std::ranges::greater Struct Reference . . . 2198
5.451.1 Detailed Description . . . 2198
5.452 std::greater< void > Struct Reference . . . 2198
5.452.1 Detailed Description . . . 2199
5.452.2 Member Typedef Documentation . . . 2199
5.453 std::greater_equal< _Tp > Struct Template Reference . . . 2199
5.453.1 Detailed Description . . . 2200
5.453.2 Member Typedef Documentation . . . 2200
5.454 std::ranges::greater_equal Struct Reference . . . 2200
5.454.1 Detailed Description . . . 2200
5.455 std::greater_equal< void > Struct Reference . . . 2201
5.455.1 Detailed Description . . . 2201
5.455.2 Member Typedef Documentation . . . 2201
5.456 __gnu_cxx::random_condition::group_adjustor Struct Reference . . . 2202
5.456.1 Detailed Description . . . 2202
5.457 __gnu_parallel::growing_blocks_tag Struct Reference . . . 2202
5.457.1 Detailed Description . . . 2202
5.458 std::gslice Class Reference . . . 2203
5.458.1 Detailed Description . . . 2203
5.459 std::gslice_array< _Tp > Class Template Reference . . . 2203
5.459.1 Detailed Description . . . 2204
5.459.2 Member Function Documentation . . . 2204
5.460 std::has_virtual_destructor< _Tp > Struct Template Reference . . . 2206
5.460.1 Detailed Description . . . 2206
5.461 std::hash< _Tp > Struct Template Reference . . . 2206
5.461.1 Detailed Description . . . 2206
5.462 std::hash< __debug::bitset< _Nb > > Struct Template Reference . . . 2206
5.462.1 Detailed Description . . . 2206
5.463 std::hash< __debug::vector< bool, _Alloc > > Struct Template Reference . . . 2207
5.463.1 Detailed Description . . . 2207
5.464 std::hash< __gnu_cxx::__u16vstring > Struct Reference . . . 2207
5.464.1 Detailed Description . . . 2207

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

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

5.503 std::hash<::bitset< _Nb > > Struct Template Reference 2220
5.503.1 Detailed Description 2220
5.504 std::hash<::vector< bool, _Alloc > > Struct Template Reference 2220
5.504.1 Detailed Description 2220
5.505 __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, Store_Hash > Struct Template Reference . . . 2221
5.505.1 Detailed Description 2221
5.506 __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, false > Struct Template Reference 2221
5.506.1 Detailed Description 2222
5.507 __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, true > Struct Template Reference 2222
5.507.1 Detailed Description 2222
5.508 __gnu_pbds::hash_exponential_size_policy< Size_Type > Class Template Reference 2222
5.508.1 Detailed Description 2223
5.508.2 Constructor & Destructor Documentation 2223
5.509 __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type > Class Template Reference 2223
5.509.1 Detailed Description 2224
5.509.2 Member Enumeration Documentation 2225
5.509.3 Constructor & Destructor Documentation 2225
5.509.4 Member Function Documentation 2225
5.510 __gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, Hold_Size > Class Template Reference 2226
5.510.1 Detailed Description 2226
5.511 __gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, true > Class Template Reference 2226
5.511.1 Detailed Description 2227
5.512 __gnu_cxx::hash_map< _Key, _Tp, _HashFn, _EqualKey, _Alloc > Class Template Reference 2227
5.512.1 Detailed Description 2228
5.513 __gnu_cxx::hash_multimap< _Key, _Tp, _HashFn, _EqualKey, _Alloc > Class Template Reference . . 2229
5.513.1 Detailed Description 2230
5.514 __gnu_cxx::hash_multiset< _Value, _HashFcn, _EqualKey, _Alloc > Class Template Reference . . . 2230
5.514.1 Detailed Description 2231
5.515 __gnu_pbds::hash_prime_size_policy Class Reference 2232
5.515.1 Detailed Description 2232
5.515.2 Member Typedef Documentation 2232
5.515.3 Constructor & Destructor Documentation 2232
5.516 __gnu_cxx::hash_set< _Value, _HashFcn, _EqualKey, _Alloc > Class Template Reference 2232
5.516.1 Detailed Description 2234
5.517 __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type > Class Template Reference 2234
5.517.1 Detailed Description 2235

5.517.2 Member Enumeration Documentation . . . 2235
5.517.3 Constructor & Destructor Documentation . . . 2236
5.517.4 Member Function Documentation . . . 2236
5.518 std::chrono::hh_mm_ss< _Duration > Class Template Reference . . . 2238
5.518.1 Detailed Description . . . 2238
5.519 std::locale::id Class Reference . . . 2238
5.519.1 Detailed Description . . . 2239
5.519.2 Constructor & Destructor Documentation . . . 2239
5.519.3 Friends And Related Symbol Documentation . . . 2239
5.520 std::thread::id Class Reference . . . 2240
5.520.1 Detailed Description . . . 2240
5.521 std::identity Struct Reference . . . 2241
5.521.1 Detailed Description . . . 2241
5.522 std::experimental::fundamentals_v1::in_place_t Struct Reference . . . 2241
5.522.1 Detailed Description . . . 2241
5.523 std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > Class Template Reference 2241
5.523.1 Detailed Description . . . 2242
5.523.2 Member Typedef Documentation . . . 2242
5.523.3 Constructor & Destructor Documentation . . . 2242
5.523.4 Member Function Documentation . . . 2243
5.523.5 Friends And Related Symbol Documentation . . . 2244
5.524 std::indirect_array< _Tp > Class Template Reference . . . 2245
5.524.1 Detailed Description . . . 2246
5.524.2 Member Function Documentation . . . 2247
5.525 std::initializer_list< _E > Class Template Reference . . . 2248
5.525.1 Detailed Description . . . 2248
5.525.2 Friends And Related Symbol Documentation . . . 2249
5.526 std::input_iterator_tag Struct Reference . . . 2249
5.526.1 Detailed Description . . . 2250
5.527 __gnu_pbds::insert_error Struct Reference . . . 2250
5.527.1 Detailed Description . . . 2251
5.527.2 Member Function Documentation . . . 2251
5.528 std::insert_iterator< _Container > Class Template Reference . . . 2251
5.528.1 Detailed Description . . . 2252
5.528.2 Member Typedef Documentation . . . 2253
5.528.3 Constructor & Destructor Documentation . . . 2253
5.528.4 Member Function Documentation . . . 2253
5.529 std::integer_sequence< _Tp, _Idx > Struct Template Reference . . . 2254
5.529.1 Detailed Description . . . 2254

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

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

5.565.1 Detailed Description . . . 2278
5.566 std::is_lvalue_reference< typename > Struct Template Reference . . . 2279
5.566.1 Detailed Description . . . 2279
5.567 std::is_member_function_pointer< _Tp > Struct Template Reference . . . 2279
5.567.1 Detailed Description . . . 2279
5.568 std::is_member_object_pointer< _Tp > Struct Template Reference . . . 2279
5.568.1 Detailed Description . . . 2279
5.569 std::is_member_pointer< _Tp > Struct Template Reference . . . 2279
5.569.1 Detailed Description . . . 2279
5.570 std::is_move_assignable< _Tp > Struct Template Reference . . . 2280
5.570.1 Detailed Description . . . 2280
5.571 std::is_move_constructible< _Tp > Struct Template Reference . . . 2280
5.571.1 Detailed Description . . . 2280
5.572 std::is_nothrow_assignable< _Tp, _Up > Struct Template Reference . . . 2280
5.572.1 Detailed Description . . . 2280
5.573 std::is_nothrow_constructible< _Tp, _Args > Struct Template Reference . . . 2280
5.573.1 Detailed Description . . . 2280
5.574 std::is_nothrow_copy_assignable< _Tp > Struct Template Reference . . . 2280
5.574.1 Detailed Description . . . 2281
5.575 std::is_nothrow_copy_constructible< _Tp > Struct Template Reference . . . 2281
5.575.1 Detailed Description . . . 2281
5.576 std::is_nothrow_default_constructible< _Tp > Struct Template Reference . . . 2281
5.576.1 Detailed Description . . . 2281
5.577 std::is_nothrow_destructible< _Tp > Struct Template Reference . . . 2281
5.577.1 Detailed Description . . . 2281
5.578 std::is_nothrow_move_assignable< _Tp > Struct Template Reference . . . 2281
5.578.1 Detailed Description . . . 2282
5.579 std::is_nothrow_move_constructible< _Tp > Struct Template Reference . . . 2282
5.579.1 Detailed Description . . . 2282
5.580 std::is_object< _Tp > Struct Template Reference . . . 2282
5.580.1 Detailed Description . . . 2282
5.581 std::is_placeholder< _Tp > Struct Template Reference . . . 2282
5.581.1 Detailed Description . . . 2283
5.582 std::is_placeholder< _Placeholder< _Num > > Struct Template Reference . . . 2283
5.582.1 Detailed Description . . . 2284
5.583 std::is_pod< _Tp > Struct Template Reference . . . 2284
5.583.1 Detailed Description . . . 2284
5.584 std::is_pointer< _Tp > Struct Template Reference . . . 2285
5.584.1 Detailed Description . . . 2285

5.585 std::is_pointer_interconvertible_base_of< _Base, _Derived > Struct Template Reference 2285
5.585.1 Detailed Description 2285
5.586 std::is_polymorphic< _Tp > Struct Template Reference 2285
5.586.1 Detailed Description 2285
5.587 std::is_reference< _Tp > Struct Template Reference 2285
5.587.1 Detailed Description 2286
5.588 std::is_rvalue_reference< typename > Struct Template Reference 2286
5.588.1 Detailed Description 2286
5.589 std::is_same< _Tp, _Up > Struct Template Reference 2286
5.589.1 Detailed Description 2286
5.590 std::is_scalar< _Tp > Struct Template Reference 2286
5.590.1 Detailed Description 2286
5.591 std::is_signed< _Tp > Struct Template Reference 2286
5.591.1 Detailed Description 2287
5.592 std::is_standard_layout< _Tp > Struct Template Reference 2287
5.592.1 Detailed Description 2287
5.593 std::is_trivial< _Tp > Struct Template Reference 2287
5.593.1 Detailed Description 2287
5.594 std::is_trivially_assignable< _Tp, _Up > Struct Template Reference 2287
5.594.1 Detailed Description 2287
5.595 std::is_trivially_constructible< _Tp, _Args > Struct Template Reference 2287
5.595.1 Detailed Description 2288
5.596 std::is_trivially_copy_assignable< _Tp > Struct Template Reference 2288
5.596.1 Detailed Description 2288
5.597 std::is_trivially_copy_constructible< _Tp > Struct Template Reference 2288
5.597.1 Detailed Description 2288
5.598 std::is_trivially_copyable< _Tp > Struct Template Reference 2288
5.598.1 Detailed Description 2288
5.599 std::is_trivially_default_constructible< _Tp > Struct Template Reference 2288
5.599.1 Detailed Description 2289
5.600 std::is_trivially_destructible< _Tp > Struct Template Reference 2289
5.600.1 Detailed Description 2289
5.601 std::is_trivially_move_assignable< _Tp > Struct Template Reference 2289
5.601.1 Detailed Description 2289
5.602 std::is_trivially_move_constructible< _Tp > Struct Template Reference 2289
5.602.1 Detailed Description 2289
5.603 std::is_union< _Tp > Struct Template Reference 2289
5.603.1 Detailed Description 2290
5.604 std::is_unsigned< _Tp > Struct Template Reference 2290

5.604.1 Detailed Description . . . 2290
5.605 std::is_void< _Tp > Struct Template Reference . . . 2290
5.605.1 Detailed Description . . . 2290
5.606 std::is_volatile< _Tp > Struct Template Reference . . . 2290
5.606.1 Detailed Description . . . 2290
5.607 std::istream_iterator< _Tp, _CharT, _Traits, _Dist > Class Template Reference . . . 2290
5.607.1 Detailed Description . . . 2291
5.607.2 Member Typedef Documentation . . . 2292
5.607.3 Constructor & Destructor Documentation . . . 2292
5.607.4 Friends And Related Symbol Documentation . . . 2292
5.608 std::istreambuf_iterator< _CharT, _Traits > Class Template Reference . . . 2293
5.608.1 Detailed Description . . . 2294
5.608.2 Member Typedef Documentation . . . 2294
5.608.3 Constructor & Destructor Documentation . . . 2295
5.608.4 Member Function Documentation . . . 2296
5.609 __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >::iterator Struct Reference . . . 2296
5.609.1 Detailed Description . . . 2298
5.610 std::experimental::filesystem::v1::path::iterator Class Reference . . . 2298
5.610.1 Detailed Description . . . 2298
5.611 std::filesystem::path::iterator Class Reference . . . 2298
5.611.1 Detailed Description . . . 2299
5.612 std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference > Struct Template Reference . . . 2299
5.612.1 Detailed Description . . . 2300
5.612.2 Member Typedef Documentation . . . 2300
5.613 std::iterator_traits< _Iterator > Struct Template Reference . . . 2301
5.613.1 Detailed Description . . . 2301
5.614 std::iterator_traits< _Tp * > Struct Template Reference . . . 2301
5.614.1 Detailed Description . . . 2301
5.615 __gnu_pbds::join_error Struct Reference . . . 2301
5.615.1 Detailed Description . . . 2302
5.615.2 Member Function Documentation . . . 2302
5.616 __gnu_pbds::detail::left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc > Class Template Reference . . . 2302
5.616.1 Detailed Description . . . 2304
5.617 __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc > Class Template Reference . . . 2304
5.617.1 Detailed Description . . . 2305
5.617.2 Member Typedef Documentation . . . 2305
5.617.3 Constructor & Destructor Documentation . . . 2306
5.617.4 Member Function Documentation . . . 2306

5.618 __gnu_pbds::detail::left_child_next_sibling_heap_node_< _Value, _Metadata, _Alloc > Struct Template Reference . . . 2307
5.618.1 Detailed Description . . . 2308
5.619 __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > Class Template Reference . . . 2308
5.619.1 Detailed Description . . . 2309
5.619.2 Member Typedef Documentation . . . 2309
5.619.3 Constructor & Destructor Documentation . . . 2310
5.619.4 Member Function Documentation . . . 2311
5.620 std::length_error Class Reference . . . 2311
5.620.1 Detailed Description . . . 2312
5.620.2 Member Function Documentation . . . 2312
5.621 std::less< _Tp > Struct Template Reference . . . 2312
5.621.1 Detailed Description . . . 2313
5.621.2 Member Typedef Documentation . . . 2313
5.622 std::ranges::less Struct Reference . . . 2314
5.622.1 Detailed Description . . . 2314
5.623 std::less_equal< _Tp > Struct Template Reference . . . 2314
5.623.1 Detailed Description . . . 2315
5.623.2 Member Typedef Documentation . . . 2315
5.624 std::ranges::less_equal Struct Reference . . . 2315
5.624.1 Detailed Description . . . 2315
5.625 std::less_equal< void > Struct Reference . . . 2315
5.625.1 Detailed Description . . . 2316
5.625.2 Member Typedef Documentation . . . 2316
5.626 __gnu_cxx::limit_condition::limit_adjustor Struct Reference . . . 2317
5.626.1 Detailed Description . . . 2317
5.627 __gnu_cxx::limit_condition Struct Reference . . . 2317
5.627.1 Detailed Description . . . 2317
5.628 std::linear_congruential_engine< _UIntType, __a, __c, __m > Class Template Reference . . . 2317
5.628.1 Detailed Description . . . 2318
5.628.2 Member Typedef Documentation . . . 2319
5.628.3 Constructor & Destructor Documentation . . . 2319
5.628.4 Member Function Documentation . . . 2319
5.628.5 Friends And Related Symbol Documentation . . . 2321
5.628.6 Member Data Documentation . . . 2323
5.629 __gnu_pbds::linear_probe_fn< Size_Type > Class Template Reference . . . 2323
5.629.1 Detailed Description . . . 2323
5.629.2 Member Function Documentation . . . 2324
5.630 std::__debug::list< _Tp, _Allocator > Class Template Reference . . . 2324

5.630.1 Detailed Description . . . 2326
5.631 std::list< _Tp, _Alloc > Class Template Reference . . . 2326
5.631.1 Detailed Description . . . 2330
5.631.2 Constructor & Destructor Documentation . . . 2330
5.631.3 Member Function Documentation . . . 2332
5.632 __gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc > Class Template Reference . . . 2347
5.632.1 Detailed Description . . . 2348
5.632.2 Constructor & Destructor Documentation . . . 2348
5.633 __gnu_pbds::list_update_tag Struct Reference . . . 2348
5.633.1 Detailed Description . . . 2349
5.634 std::locale Class Reference . . . 2349
5.634.1 Detailed Description . . . 2350
5.634.2 Member Typedef Documentation . . . 2351
5.634.3 Constructor & Destructor Documentation . . . 2351
5.634.4 Member Function Documentation . . . 2353
5.634.5 Friends And Related Symbol Documentation . . . 2356
5.634.6 Member Data Documentation . . . 2357
5.635 std::lock_guard< _Mutex > Class Template Reference . . . 2358
5.635.1 Detailed Description . . . 2358
5.636 std::logic_error Class Reference . . . 2358
5.636.1 Detailed Description . . . 2359
5.636.2 Constructor & Destructor Documentation . . . 2359
5.636.3 Member Function Documentation . . . 2359
5.637 std::logical_and< _Tp > Struct Template Reference . . . 2360
5.637.1 Detailed Description . . . 2360
5.637.2 Member Typedef Documentation . . . 2360
5.638 std::logical_and< void > Struct Reference . . . 2361
5.638.1 Detailed Description . . . 2361
5.638.2 Member Typedef Documentation . . . 2361
5.639 std::logical_not< _Tp > Struct Template Reference . . . 2362
5.639.1 Detailed Description . . . 2362
5.639.2 Member Typedef Documentation . . . 2362
5.640 std::logical_not< void > Struct Reference . . . 2363
5.640.1 Detailed Description . . . 2363
5.640.2 Member Typedef Documentation . . . 2363
5.641 std::logical_or< _Tp > Struct Template Reference . . . 2364
5.641.1 Detailed Description . . . 2364
5.641.2 Member Typedef Documentation . . . 2364
5.642 std::logical_or< void > Struct Reference . . . 2365

5.642.1 Detailed Description . . . 2365
5.642.2 Member Typedef Documentation . . . 2365
5.643 std::lognormal_distribution< _RealType > Class Template Reference . . . 2366
5.643.1 Detailed Description . . . 2367
5.643.2 Member Typedef Documentation . . . 2367
5.643.3 Member Function Documentation . . . 2367
5.643.4 Friends And Related Symbol Documentation . . . 2368
5.644 __gnu_pbds::detail::lu_counter_metadata< Size_Type > Class Template Reference . . . 2369
5.644.1 Detailed Description . . . 2369
5.645 __gnu_pbds::lu_counter_policy< Max_Count, _Alloc > Class Template Reference . . . 2369
5.645.1 Detailed Description . . . 2370
5.645.2 Member Typedef Documentation . . . 2370
5.645.3 Member Enumeration Documentation . . . 2371
5.645.4 Member Function Documentation . . . 2371
5.646 __gnu_pbds::detail::lu_counter_policy_base< Size_Type > Class Template Reference . . . 2371
5.646.1 Detailed Description . . . 2372
5.647 __gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy > Class Template Reference 2372
5.647.1 Detailed Description . . . 2375
5.648 __gnu_pbds::lu_move_to_front_policy< _Alloc > Class Template Reference . . . 2375
5.648.1 Detailed Description . . . 2375
5.648.2 Member Typedef Documentation . . . 2375
5.648.3 Member Function Documentation . . . 2375
5.649 std::make_signed< _Tp > Struct Template Reference . . . 2376
5.649.1 Detailed Description . . . 2376
5.650 std::make_unsigned< _Tp > Struct Template Reference . . . 2376
5.650.1 Detailed Description . . . 2376
5.651 __gnu_cxx::malloc_allocator< _Tp > Class Template Reference . . . 2376
5.651.1 Detailed Description . . . 2377
5.652 std::__debug::map< _Key, _Tp, _Compare, _Allocator > Class Template Reference . . . 2377
5.652.1 Detailed Description . . . 2380
5.653 std::map< _Key, _Tp, _Compare, _Alloc > Class Template Reference . . . 2380
5.653.1 Detailed Description . . . 2383
5.653.2 Constructor & Destructor Documentation . . . 2383
5.653.3 Member Function Documentation . . . 2386
5.654 std::mask_array< _Tp > Class Template Reference . . . 2405
5.654.1 Detailed Description . . . 2406
5.654.2 Member Function Documentation . . . 2406
5.655 __gnu_pbds::detail::mask_based_range_hashing< Size_Type > Class Template Reference . . . 2408
5.655.1 Detailed Description . . . 2408

5.656 std::match_results< _Bi_iter, _Alloc > Class Template Reference 2409
5.656.1 Detailed Description 2411
5.656.2 Constructor & Destructor Documentation 2411
5.656.3 Member Function Documentation 2412
5.657 __gnu_pbds::detail::maybe_null_type< Key, Mapped, _Alloc, Store_Hash > Struct Template Reference 2417
5.657.1 Detailed Description 2418
5.658 __gnu_pbds::detail::maybe_null_type< Key, null_type, _Alloc, Store_Hash > Struct Template Reference 2419
5.658.1 Detailed Description 2419
5.659 std::mem_fun1_ref_t< _Ret, _Tp, _Arg > Class Template Reference 2419
5.659.1 Detailed Description 2420
5.659.2 Member Typedef Documentation 2420
5.660 std::mem_fun1_t< _Ret, _Tp, _Arg > Class Template Reference 2421
5.660.1 Detailed Description 2421
5.660.2 Member Typedef Documentation 2421
5.661 std::mem_fun_ref_t< _Ret, _Tp > Class Template Reference 2422
5.661.1 Detailed Description 2422
5.661.2 Member Typedef Documentation 2422
5.662 std::mem_fun_t< _Ret, _Tp > Class Template Reference 2423
5.662.1 Detailed Description 2423
5.662.2 Member Typedef Documentation 2423
5.663 std::pmr::memory_resource Class Reference 2424
5.663.1 Detailed Description 2424
5.664 std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > Class Template Reference 2424
5.664.1 Detailed Description 2426
5.664.2 Member Typedef Documentation 2426
5.664.3 Constructor & Destructor Documentation 2427
5.664.4 Member Function Documentation 2427
5.664.5 Friends And Related Symbol Documentation 2428
5.665 std::messages< _CharT > Class Template Reference 2429
5.665.1 Detailed Description 2430
5.665.2 Member Typedef Documentation 2431
5.665.3 Constructor & Destructor Documentation 2431
5.665.4 Member Function Documentation 2432
5.665.5 Member Data Documentation 2432
5.666 std::messages_base Struct Reference 2432
5.666.1 Detailed Description 2433
5.667 std::messages_byname< _CharT > Class Template Reference 2433
5.667.1 Detailed Description 2434

5.667.2 Member Function Documentation . . . 2434
5.667.3 Member Data Documentation . . . 2434
5.668 std::minus< _Tp > Struct Template Reference . . . 2435
5.668.1 Detailed Description . . . 2435
5.668.2 Member Typedef Documentation . . . 2435
5.669 std::minus< void > Struct Reference . . . 2436
5.669.1 Detailed Description . . . 2436
5.669.2 Member Typedef Documentation . . . 2436
5.670 __gnu_pbds::detail::mod_based_range_hashing< Size_Type > Class Template Reference . . . 2437
5.670.1 Detailed Description . . . 2437
5.671 std::modulus< _Tp > Struct Template Reference . . . 2437
5.671.1 Detailed Description . . . 2438
5.671.2 Member Typedef Documentation . . . 2438
5.672 std::modulus< void > Struct Reference . . . 2438
5.672.1 Detailed Description . . . 2439
5.672.2 Member Typedef Documentation . . . 2439
5.673 std::money_base Class Reference . . . 2439
5.673.1 Detailed Description . . . 2440
5.674 std::money_get< _CharT, _InIter > Class Template Reference . . . 2441
5.674.1 Detailed Description . . . 2442
5.674.2 Member Typedef Documentation . . . 2442
5.674.3 Constructor & Destructor Documentation . . . 2442
5.674.4 Member Function Documentation . . . 2443
5.674.5 Member Data Documentation . . . 2445
5.675 std::money_put< _CharT, _OutIter > Class Template Reference . . . 2445
5.675.1 Detailed Description . . . 2446
5.675.2 Member Typedef Documentation . . . 2446
5.675.3 Constructor & Destructor Documentation . . . 2447
5.675.4 Member Function Documentation . . . 2447
5.675.5 Member Data Documentation . . . 2449
5.676 std::moneypunct< _CharT, _Intl > Class Template Reference . . . 2449
5.676.1 Detailed Description . . . 2451
5.676.2 Member Typedef Documentation . . . 2451
5.676.3 Constructor & Destructor Documentation . . . 2452
5.676.4 Member Function Documentation . . . 2453
5.676.5 Member Data Documentation . . . 2458
5.677 std::moneypunct_byname< _CharT, _Intl > Class Template Reference . . . 2458
5.677.1 Detailed Description . . . 2460
5.677.2 Member Function Documentation . . . 2460

5.677.3 Member Data Documentation . . . 2466
5.678 std::pmr::monotonic_buffer_resource Class Reference . . . 2466
5.678.1 Detailed Description . . . 2467
5.678.2 Member Function Documentation . . . 2467
5.679 std::move_iterator< _Iterator > Class Template Reference . . . 2467
5.679.1 Detailed Description . . . 2468
5.680 std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> Class Template Reference . . . 2469
5.680.1 Detailed Description . . . 2469
5.680.2 Constructor & Destructor Documentation . . . 2470
5.680.3 Member Function Documentation . . . 2471
5.680.4 Friends And Related Symbol Documentation . . . 2472
5.681 std::move_sentinel< _Sent > Class Template Reference . . . 2473
5.681.1 Detailed Description . . . 2473
5.682 std::__debug::multimap< _Key, _Tp, _Compare, _Allocator > Class Template Reference . . . 2473
5.682.1 Detailed Description . . . 2476
5.683 std::multimap< _Key, _Tp, _Compare, _Alloc > Class Template Reference . . . 2476
5.683.1 Detailed Description . . . 2479
5.683.2 Constructor & Destructor Documentation . . . 2479
5.683.3 Member Function Documentation . . . 2482
5.684 std::multiplies< _Tp > Struct Template Reference . . . 2499
5.684.1 Detailed Description . . . 2500
5.684.2 Member Typedef Documentation . . . 2500
5.685 std::multiplies< void > Struct Reference . . . 2500
5.685.1 Detailed Description . . . 2501
5.685.2 Member Typedef Documentation . . . 2501
5.686 std::__debug::multiset< _Key, _Compare, _Allocator > Class Template Reference . . . 2501
5.686.1 Detailed Description . . . 2504
5.687 std::multiset< _Key, _Compare, _Alloc > Class Template Reference . . . 2504
5.687.1 Detailed Description . . . 2506
5.687.2 Constructor & Destructor Documentation . . . 2507
5.687.3 Member Function Documentation . . . 2510
5.688 __gnu_parallel::multiway_mergesort_exact_tag Struct Reference . . . 2524
5.688.1 Detailed Description . . . 2524
5.688.2 Member Function Documentation . . . 2524
5.689 __gnu_parallel::multiway_mergesort_sampling_tag Struct Reference . . . 2525
5.689.1 Detailed Description . . . 2525
5.689.2 Member Function Documentation . . . 2525
5.690 __gnu_parallel::multiway_mergesort_tag Struct Reference . . . 2526

5.690.1 Detailed Description . . . 2526
5.690.2 Member Function Documentation . . . 2526
5.691 std::mutex Class Reference . . . 2527
5.691.1 Detailed Description . . . 2527
5.692 std::negate< _Tp > Struct Template Reference . . . 2527
5.692.1 Detailed Description . . . 2528
5.692.2 Member Typedef Documentation . . . 2528
5.693 std::negate< void > Struct Reference . . . 2528
5.693.1 Detailed Description . . . 2529
5.693.2 Member Typedef Documentation . . . 2529
5.694 std::negative_binomial_distribution< _IntType > Class Template Reference . . . 2529
5.694.1 Detailed Description . . . 2530
5.694.2 Member Typedef Documentation . . . 2531
5.694.3 Member Function Documentation . . . 2531
5.694.4 Friends And Related Symbol Documentation . . . 2532
5.695 std::nested_exception Class Reference . . . 2533
5.695.1 Detailed Description . . . 2533
5.695.2 Constructor & Destructor Documentation . . . 2533
5.695.3 Member Function Documentation . . . 2533
5.696 __gnu_cxx::limit_condition::never_adjustor Struct Reference . . . 2534
5.696.1 Detailed Description . . . 2534
5.697 __gnu_cxx::random_condition::never_adjustor Struct Reference . . . 2534
5.697.1 Detailed Description . . . 2534
5.698 __gnu_cxx::new_allocator< _Tp > Class Template Reference . . . 2534
5.698.1 Detailed Description . . . 2535
5.699 __gnu_pbds::detail::no_throw_copies< Key, Mapped > Struct Template Reference . . . 2535
5.699.1 Detailed Description . . . 2535
5.700 __gnu_pbds::detail::no_throw_copies< Key, null_type > Struct Template Reference . . . 2536
5.700.1 Detailed Description . . . 2536
5.701 std::normal_distribution< _RealType > Class Template Reference . . . 2536
5.701.1 Detailed Description . . . 2537
5.701.2 Member Typedef Documentation . . . 2537
5.701.3 Constructor & Destructor Documentation . . . 2537
5.701.4 Member Function Documentation . . . 2537
5.701.5 Friends And Related Symbol Documentation . . . 2539
5.702 std::not_equal_to< _Tp > Struct Template Reference . . . 2540
5.702.1 Detailed Description . . . 2540
5.702.2 Member Typedef Documentation . . . 2540
5.703 std::ranges::not_equal_to Struct Reference . . . 2541

5.703.1 Detailed Description . . . 2541
5.704 std::not_equal_to< void > Struct Reference . . . 2541
5.704.1 Detailed Description . . . 2542
5.704.2 Member Typedef Documentation . . . 2542
5.705 __gnu_pbds::null_node_update< _Tp1, _Tp2, _Tp3, _Tp4 > Struct Template Reference . . . 2542
5.705.1 Detailed Description . . . 2543
5.706 __gnu_pbds::null_type Struct Reference . . . 2543
5.706.1 Detailed Description . . . 2544
5.707 std::experimental::fundamentals_v1::nullopt_t Struct Reference . . . 2544
5.707.1 Detailed Description . . . 2545
5.708 std::num_get< _CharT, _InIter > Class Template Reference . . . 2545
5.708.1 Detailed Description . . . 2547
5.708.2 Member Typedef Documentation . . . 2547
5.708.3 Constructor & Destructor Documentation . . . 2547
5.708.4 Member Function Documentation . . . 2547
5.708.5 Member Data Documentation . . . 2560
5.709 std::num_put< _CharT, _OutIter > Class Template Reference . . . 2560
5.709.1 Detailed Description . . . 2562
5.709.2 Member Typedef Documentation . . . 2562
5.709.3 Constructor & Destructor Documentation . . . 2563
5.709.4 Member Function Documentation . . . 2563
5.709.5 Member Data Documentation . . . 2573
5.710 std::numeric_limits< _Tp > Struct Template Reference . . . 2573
5.710.1 Detailed Description . . . 2574
5.710.2 Member Function Documentation . . . 2574
5.710.3 Member Data Documentation . . . 2575
5.711 std::numeric_limits< bool > Struct Reference . . . 2578
5.711.1 Detailed Description . . . 2580
5.711.2 Member Function Documentation . . . 2580
5.711.3 Member Data Documentation . . . 2581
5.712 std::numeric_limits< char > Struct Reference . . . 2583
5.712.1 Detailed Description . . . 2585
5.712.2 Member Function Documentation . . . 2585
5.712.3 Member Data Documentation . . . 2586
5.713 std::numeric_limits< char16_t > Struct Reference . . . 2588
5.713.1 Detailed Description . . . 2590
5.713.2 Member Function Documentation . . . 2590
5.713.3 Member Data Documentation . . . 2591
5.714 std::numeric_limits< char32_t > Struct Reference . . . 2594

5.714.1 Detailed Description . . . 2595
5.714.2 Member Function Documentation . . . 2596
5.714.3 Member Data Documentation . . . 2597
5.715 std::numeric_limits< double > Struct Reference . . . 2599
5.715.1 Detailed Description . . . 2601
5.715.2 Member Function Documentation . . . 2601
5.715.3 Member Data Documentation . . . 2602
5.716 std::numeric_limits< float > Struct Reference . . . 2604
5.716.1 Detailed Description . . . 2606
5.716.2 Member Function Documentation . . . 2606
5.716.3 Member Data Documentation . . . 2607
5.717 std::numeric_limits< int > Struct Reference . . . 2609
5.717.1 Detailed Description . . . 2611
5.717.2 Member Function Documentation . . . 2611
5.717.3 Member Data Documentation . . . 2612
5.718 std::numeric_limits< long > Struct Reference . . . 2615
5.718.1 Detailed Description . . . 2616
5.718.2 Member Function Documentation . . . 2616
5.718.3 Member Data Documentation . . . 2617
5.719 std::numeric_limits< long double > Struct Reference . . . 2620
5.719.1 Detailed Description . . . 2622
5.719.2 Member Function Documentation . . . 2622
5.719.3 Member Data Documentation . . . 2623
5.720 std::numeric_limits< long long > Struct Reference . . . 2625
5.720.1 Detailed Description . . . 2627
5.720.2 Member Function Documentation . . . 2627
5.720.3 Member Data Documentation . . . 2628
5.721 std::numeric_limits< short > Struct Reference . . . 2630
5.721.1 Detailed Description . . . 2632
5.721.2 Member Function Documentation . . . 2632
5.721.3 Member Data Documentation . . . 2633
5.722 std::numeric_limits< signed char > Struct Reference . . . 2636
5.722.1 Detailed Description . . . 2637
5.722.2 Member Function Documentation . . . 2638
5.722.3 Member Data Documentation . . . 2639
5.723 std::numeric_limits< unsigned char > Struct Reference . . . 2641
5.723.1 Detailed Description . . . 2643
5.723.2 Member Function Documentation . . . 2643
5.723.3 Member Data Documentation . . . 2644

5.724 std::numeric_limits< unsigned int > Struct Reference 2646
5.724.1 Detailed Description 2648
5.724.2 Member Function Documentation 2648
5.724.3 Member Data Documentation 2649
5.725 std::numeric_limits< unsigned long > Struct Reference 2651
5.725.1 Detailed Description 2653
5.725.2 Member Function Documentation 2653
5.725.3 Member Data Documentation 2654
5.726 std::numeric_limits< unsigned long long > Struct Reference 2657
5.726.1 Detailed Description 2658
5.726.2 Member Function Documentation 2659
5.726.3 Member Data Documentation 2660
5.727 std::numeric_limits< unsigned short > Struct Reference 2662
5.727.1 Detailed Description 2664
5.727.2 Member Function Documentation 2664
5.727.3 Member Data Documentation 2665
5.728 std::numeric_limits< wchar_t > Struct Reference 2667
5.728.1 Detailed Description 2669
5.728.2 Member Function Documentation 2669
5.728.3 Member Data Documentation 2670
5.729 std::numpunct< _CharT > Class Template Reference 2672
5.729.1 Detailed Description 2674
5.729.2 Member Typedef Documentation 2674
5.729.3 Constructor & Destructor Documentation 2674
5.729.4 Member Function Documentation 2675
5.729.5 Member Data Documentation 2678
5.730 std::numpunct_byname< _CharT > Class Template Reference 2678
5.730.1 Detailed Description 2679
5.730.2 Member Function Documentation 2679
5.730.3 Member Data Documentation 2682
5.731 __gnu_parallel::omp_loop_static_tag Struct Reference 2682
5.731.1 Detailed Description 2683
5.731.2 Member Function Documentation 2683
5.732 __gnu_parallel::omp_loop_tag Struct Reference 2683
5.732.1 Detailed Description 2684
5.732.2 Member Function Documentation 2684
5.733 std::once_flag Struct Reference 2684
5.733.1 Detailed Description 2684
5.733.2 Constructor & Destructor Documentation 2685

5.733.3 Member Function Documentation . . . 2685
5.733.4 Friends And Related Symbol Documentation . . . 2685
5.734 std::experimental::fundamentals_v1::optional< _Tp > Class Template Reference . . . 2685
5.734.1 Detailed Description . . . 2686
5.735 std::ostream_iterator< _Tp, _CharT, _Traits > Class Template Reference . . . 2687
5.735.1 Detailed Description . . . 2687
5.735.2 Member Typedef Documentation . . . 2688
5.735.3 Constructor & Destructor Documentation . . . 2689
5.735.4 Member Function Documentation . . . 2689
5.736 std::experimental::fundamentals_v2::ostream_joiner< _DelimT, _CharT, _Traits > Class Template Reference . . . 2689
5.736.1 Detailed Description . . . 2690
5.737 std::ostreambuf_iterator< _CharT, _Traits > Class Template Reference . . . 2690
5.737.1 Detailed Description . . . 2691
5.737.2 Member Typedef Documentation . . . 2691
5.737.3 Constructor & Destructor Documentation . . . 2692
5.737.4 Member Function Documentation . . . 2692
5.738 std::out_of_range Class Reference . . . 2693
5.738.1 Detailed Description . . . 2694
5.738.2 Member Function Documentation . . . 2694
5.739 std::output_iterator_tag Struct Reference . . . 2694
5.739.1 Detailed Description . . . 2695
5.740 __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > Class Template Reference . . . 2695
5.740.1 Detailed Description . . . 2697
5.740.2 Member Function Documentation . . . 2697
5.741 __gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc > Class Template Reference . . . 2698
5.741.1 Detailed Description . . . 2699
5.741.2 Member Function Documentation . . . 2699
5.742 __gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc > Class Template Reference . . . 2700
5.742.1 Detailed Description . . . 2701
5.742.2 Member Function Documentation . . . 2701
5.743 __gnu_pbds::ov_tree_tag Struct Reference . . . 2701
5.743.1 Detailed Description . . . 2702
5.744 std::overflow_error Class Reference . . . 2702
5.744.1 Detailed Description . . . 2703
5.744.2 Member Function Documentation . . . 2703
5.745 std::owner_less< _Tp > Struct Template Reference . . . 2703
5.745.1 Detailed Description . . . 2703

5.746 std::experimental::fundamentals_v2::owner_less< shared_ptr< _Tp > > Struct Template Reference . 2704
5.746.1 Detailed Description . . . 2704
5.746.2 Member Typedef Documentation . . . 2704
5.747 std::owner_less< shared_ptr< _Tp > > Struct Template Reference . . . 2704
5.747.1 Detailed Description . . . 2705
5.747.2 Member Typedef Documentation . . . 2705
5.748 std::owner_less< void > Struct Reference . . . 2705
5.748.1 Detailed Description . . . 2705
5.748.2 Member Typedef Documentation . . . 2706
5.749 std::experimental::fundamentals_v2::owner_less< weak_ptr< _Tp > > Struct Template Reference . . 2706
5.749.1 Detailed Description . . . 2706
5.749.2 Member Typedef Documentation . . . 2706
5.750 std::owner_less< weak_ptr< _Tp > > Struct Template Reference . . . 2707
5.750.1 Detailed Description . . . 2707
5.750.2 Member Typedef Documentation . . . 2707
5.751 std::packaged_task< _Res(_ArgTypes...)> Class Template Reference . . . 2707
5.751.1 Detailed Description . . . 2708
5.752 __gnu_cxx::pair< _T1, _T2 > Struct Template Reference . . . 2708
5.752.1 Detailed Description . . . 2710
5.752.2 Member Typedef Documentation . . . 2710
5.752.3 Constructor & Destructor Documentation . . . 2710
5.752.4 Member Function Documentation . . . 2711
5.752.5 Member Data Documentation . . . 2711
5.753 std::pair< _T1, _T2 > Struct Template Reference . . . 2711
5.753.1 Detailed Description . . . 2713
5.753.2 Member Typedef Documentation . . . 2713
5.753.3 Constructor & Destructor Documentation . . . 2714
5.753.4 Member Function Documentation . . . 2715
5.753.5 Member Data Documentation . . . 2715
5.754 __gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc > Class Template Reference . . . 2715
5.754.1 Detailed Description . . . 2717
5.755 __gnu_pbds::pairing_heap_tag Struct Reference . . . 2718
5.755.1 Detailed Description . . . 2718
5.756 __gnu_parallel::parallel_tag Struct Reference . . . 2718
5.756.1 Detailed Description . . . 2720
5.756.2 Constructor & Destructor Documentation . . . 2720
5.756.3 Member Function Documentation . . . 2720
5.757 std::bernoulli_distribution::param_type Struct Reference . . . 2720
5.757.1 Detailed Description . . . 2721

5.758 std::binomial_distribution< _IntType >::param_type Struct Reference 2721
5.758.1 Detailed Description 2721
5.759 std::cauchy_distribution< _RealType >::param_type Struct Reference 2721
5.759.1 Detailed Description 2722
5.760 std::chi_squared_distribution< _RealType >::param_type Struct Reference 2722
5.760.1 Detailed Description 2722
5.761 std::discrete_distribution< _IntType >::param_type Struct Reference 2722
5.761.1 Detailed Description 2723
5.762 std::exponential_distribution< _RealType >::param_type Struct Reference 2723
5.762.1 Detailed Description 2723
5.763 std::extreme_value_distribution< _RealType >::param_type Struct Reference 2723
5.763.1 Detailed Description 2724
5.764 std::fisher_f_distribution< _RealType >::param_type Struct Reference 2724
5.764.1 Detailed Description 2724
5.765 std::gamma_distribution< _RealType >::param_type Struct Reference 2724
5.765.1 Detailed Description 2725
5.766 std::geometric_distribution< _IntType >::param_type Struct Reference 2725
5.766.1 Detailed Description 2725
5.767 std::lognormal_distribution< _RealType >::param_type Struct Reference 2725
5.767.1 Detailed Description 2726
5.768 std::negative_binomial_distribution< _IntType >::param_type Struct Reference 2726
5.768.1 Detailed Description 2726
5.769 std::normal_distribution< _RealType >::param_type Struct Reference 2726
5.769.1 Detailed Description 2727
5.770 std::piecewise_constant_distribution< _RealType >::param_type Struct Reference 2727
5.770.1 Detailed Description 2728
5.771 std::piecewise_linear_distribution< _RealType >::param_type Struct Reference 2728
5.771.1 Detailed Description 2728
5.772 std::poisson_distribution< _IntType >::param_type Struct Reference 2728
5.772.1 Detailed Description 2729
5.773 std::student_t_distribution< _RealType >::param_type Struct Reference 2729
5.773.1 Detailed Description 2729
5.774 std::uniform_int_distribution< _IntType >::param_type Struct Reference 2729
5.774.1 Detailed Description 2730
5.775 std::uniform_real_distribution< _RealType >::param_type Struct Reference 2730
5.775.1 Detailed Description 2730
5.776 std::weibull_distribution< _RealType >::param_type Struct Reference 2730
5.776.1 Detailed Description 2731
5.777 __gnu_pbds::detail::pat_trie_base Struct Reference 2731

5.777.1 Detailed Description . . . 2732
5.777.2 Member Enumeration Documentation . . . 2732
5.778 __gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc > Class Template Reference . . . 2732
5.778.1 Detailed Description . . . 2734
5.778.2 Member Enumeration Documentation . . . 2734
5.778.3 Member Function Documentation . . . 2734
5.779 __gnu_pbds::pat_trie_tag Struct Reference . . . 2735
5.779.1 Detailed Description . . . 2736
5.780 std::experimental::filesystem::v1::path Class Reference . . . 2736
5.780.1 Detailed Description . . . 2738
5.781 std::filesystem::path Class Reference . . . 2739
5.781.1 Detailed Description . . . 2741
5.781.2 Member Enumeration Documentation . . . 2741
5.781.3 Friends And Related Symbol Documentation . . . 2742
5.782 std::piecewise_constant_distribution< _RealType > Class Template Reference . . . 2742
5.782.1 Detailed Description . . . 2743
5.782.2 Member Typedef Documentation . . . 2744
5.782.3 Member Function Documentation . . . 2744
5.782.4 Friends And Related Symbol Documentation . . . 2746
5.783 std::piecewise_construct_t Struct Reference . . . 2747
5.783.1 Detailed Description . . . 2747
5.784 std::piecewise_linear_distribution< _RealType > Class Template Reference . . . 2747
5.784.1 Detailed Description . . . 2748
5.784.2 Member Typedef Documentation . . . 2748
5.784.3 Member Function Documentation . . . 2748
5.784.4 Friends And Related Symbol Documentation . . . 2749
5.785 std::plus< _Tp > Struct Template Reference . . . 2750
5.785.1 Detailed Description . . . 2751
5.785.2 Member Typedef Documentation . . . 2751
5.785.3 Member Function Documentation . . . 2752
5.786 __gnu_pbds::point_invalidation_guarantee Struct Reference . . . 2752
5.786.1 Detailed Description . . . 2752
5.787 std::pointer_to_binary_function< _Arg1, _Arg2, _Result > Class Template Reference . . . 2752
5.787.1 Detailed Description . . . 2753
5.787.2 Member Typedef Documentation . . . 2753
5.788 std::pointer_to_unary_function< _Arg, _Result > Class Template Reference . . . 2754
5.788.1 Detailed Description . . . 2754
5.788.2 Member Typedef Documentation . . . 2755

5.789 std::pointer_traits< _Ptr > Struct Template Reference 2755
5.789.1 Detailed Description 2755
5.789.2 Member Typedef Documentation 2755
5.789.3 Member Function Documentation 2756
5.790 std::pointer_traits< _Tp * > Struct Template Reference 2756
5.790.1 Detailed Description 2757
5.790.2 Member Typedef Documentation 2757
5.790.3 Member Function Documentation 2758
5.791 std::poisson_distribution< _IntType > Class Template Reference 2759
5.791.1 Detailed Description 2760
5.791.2 Member Typedef Documentation 2760
5.791.3 Member Function Documentation 2760
5.791.4 Friends And Related Symbol Documentation 2761
5.792 std::pmr::polymorphic_allocator< _Tp > Class Template Reference 2762
5.792.1 Detailed Description 2763
5.793 std::pmr::pool_options Struct Reference 2763
5.793.1 Detailed Description 2763
5.793.2 Member Data Documentation 2764
5.794 __gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc > Class Template Reference 2764
5.794.1 Detailed Description 2764
5.794.2 Constructor & Destructor Documentation 2765
5.795 std::priority_queue< _Tp, _Sequence, _Compare > Class Template Reference 2765
5.795.1 Detailed Description 2767
5.795.2 Constructor & Destructor Documentation 2767
5.795.3 Member Function Documentation 2768
5.796 __gnu_pbds::priority_queue_tag Struct Reference 2769
5.796.1 Detailed Description 2769
5.797 __gnu_pbds::detail::probe_fn_base< _Alloc > Class Template Reference 2770
5.797.1 Detailed Description 2770
5.798 __gnu_cxx::project1st< _Arg1, _Arg2 > Struct Template Reference 2770
5.798.1 Detailed Description 2770
5.798.2 Member Typedef Documentation 2770
5.799 __gnu_cxx::project2nd< _Arg1, _Arg2 > Struct Template Reference 2770
5.799.1 Detailed Description 2771
5.799.2 Member Typedef Documentation 2771
5.800 std::promise< _Res > Class Template Reference 2771
5.800.1 Detailed Description 2772
5.801 std::promise< _Res & > Class Template Reference 2772
5.801.1 Detailed Description 2773

5.802 std::promise< void > Class Reference . . . 2773
5.802.1 Detailed Description . . . 2774
5.803 std::experimental::fundamentals_v2::propagate_const< _Tp > Class Template Reference . . . 2774
5.803.1 Detailed Description . . . 2775
5.804 __gnu_pbds::quadratic_probe_fn< Size_Type > Class Template Reference . . . 2775
5.804.1 Detailed Description . . . 2775
5.804.2 Member Function Documentation . . . 2775
5.805 std::queue< _Tp, _Sequence > Class Template Reference . . . 2775
5.805.1 Detailed Description . . . 2777
5.805.2 Constructor & Destructor Documentation . . . 2777
5.805.3 Member Function Documentation . . . 2777
5.805.4 Member Data Documentation . . . 2778
5.806 __gnu_parallel::quicksort_tag Struct Reference . . . 2779
5.806.1 Detailed Description . . . 2779
5.806.2 Member Function Documentation . . . 2779
5.807 std::random_access_iterator_tag Struct Reference . . . 2780
5.807.1 Detailed Description . . . 2780
5.808 __gnu_cxx::random_condition Struct Reference . . . 2780
5.808.1 Detailed Description . . . 2781
5.809 std::random_device Class Reference . . . 2781
5.809.1 Detailed Description . . . 2782
5.809.2 Member Typedef Documentation . . . 2782
5.810 std::range_error Class Reference . . . 2782
5.810.1 Detailed Description . . . 2782
5.810.2 Member Function Documentation . . . 2783
5.811 __gnu_pbds::range_invalidation_guarantee Struct Reference . . . 2783
5.811.1 Detailed Description . . . 2783
5.812 __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash > Class Template Reference . . . 2784
5.812.1 Detailed Description . . . 2784
5.813 __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, false > Class Template Reference . . . 2784
5.813.1 Detailed Description . . . 2785
5.814 __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true > Class Template Reference . . . 2785
5.814.1 Detailed Description . . . 2785
5.815 __gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, false > Class Template Reference . . . 2785
5.815.1 Detailed Description . . . 2786

5.816 __gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, true > Class Template Reference . . . 2786
5.816.1 Detailed Description . . . 2786
5.817 __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_Hash > Class Template Reference . . . 2787
5.817.1 Detailed Description . . . 2787
5.818 __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, false > Class Template Reference . . . 2787
5.818.1 Detailed Description . . . 2788
5.819 __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true > Class Template Reference . . . 2788
5.819.1 Detailed Description . . . 2788
5.820 __gnu_pbds::detail::ranged_probe_fn< Key, null_type, _Alloc, Comb_Probe_Fn, null_type, false > Class Template Reference . . . 2789
5.820.1 Detailed Description . . . 2789
5.821 std::rank< _Tp > Struct Template Reference . . . 2789
5.821.1 Detailed Description . . . 2790
5.822 std::ratio< _Num, _Den > Struct Template Reference . . . 2790
5.822.1 Detailed Description . . . 2790
5.823 std::ratio_equal< _R1, _R2 > Struct Template Reference . . . 2790
5.823.1 Detailed Description . . . 2791
5.824 std::ratio_greater< _R1, _R2 > Struct Template Reference . . . 2791
5.824.1 Detailed Description . . . 2792
5.825 std::ratio_greater_equal< _R1, _R2 > Struct Template Reference . . . 2792
5.825.1 Detailed Description . . . 2793
5.826 std::ratio_less< _R1, _R2 > Struct Template Reference . . . 2793
5.826.1 Detailed Description . . . 2793
5.827 std::ratio_less_equal< _R1, _R2 > Struct Template Reference . . . 2794
5.827.1 Detailed Description . . . 2794
5.828 std::ratio_not_equal< _R1, _R2 > Struct Template Reference . . . 2794
5.828.1 Detailed Description . . . 2795
5.829 std::raw_storage_iterator< _OutputIterator, _Tp > Class Template Reference . . . 2795
5.829.1 Detailed Description . . . 2796
5.829.2 Member Typedef Documentation . . . 2796
5.830 __gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc > Struct Template Reference . . . 2797
5.830.1 Detailed Description . . . 2800
5.831 __gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > Class Template Reference . . . 2801
5.831.1 Detailed Description . . . 2803
5.831.2 Member Function Documentation . . . 2804
5.832 __gnu_pbds::detail::rb_tree_node_< Value_Type, Metadata, _Alloc > Struct Template Reference . . . 2804

5.832.1 Detailed Description . . . 2805
5.833 __gnu_pbds::rb_tree_tag Struct Reference . . . 2805
5.833.1 Detailed Description . . . 2806
5.834 __gnu_pbds::detail::rc< _Node, _Alloc > Class Template Reference . . . 2806
5.834.1 Detailed Description . . . 2807
5.835 __gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc > Class Template Reference . . . 2807
5.835.1 Detailed Description . . . 2809
5.836 __gnu_pbds::rc_binomial_heap_tag Struct Reference . . . 2809
5.836.1 Detailed Description . . . 2809
5.837 __gnu_pbds::detail::rebind_traits< _Alloc, T > Struct Template Reference . . . 2809
5.837.1 Detailed Description . . . 2810
5.838 std::filesystem::recursive_directory_iterator Class Reference . . . 2810
5.838.1 Detailed Description . . . 2811
5.839 __gnu_cxx::recursive_init_error Class Reference . . . 2811
5.839.1 Detailed Description . . . 2811
5.839.2 Member Function Documentation . . . 2811
5.840 std::recursive_mutex Class Reference . . . 2812
5.840.1 Detailed Description . . . 2812
5.841 std::recursive_timed_mutex Class Reference . . . 2812
5.841.1 Detailed Description . . . 2813
5.842 std::bitset< _Nb >::reference Class Reference . . . 2813
5.842.1 Detailed Description . . . 2813
5.843 std::tr2::dynamic_bitset< _WordT, _Alloc >::reference Class Reference . . . 2813
5.843.1 Detailed Description . . . 2814
5.844 std::reference_wrapper< _Tp > Class Template Reference . . . 2814
5.844.1 Detailed Description . . . 2814
5.844.2 Friends And Related Symbol Documentation . . . 2815
5.845 std::regex_error Class Reference . . . 2815
5.845.1 Detailed Description . . . 2816
5.845.2 Constructor & Destructor Documentation . . . 2816
5.845.3 Member Function Documentation . . . 2816
5.846 std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > Class Template Reference . . . 2817
5.846.1 Detailed Description . . . 2817
5.846.2 Constructor & Destructor Documentation . . . 2818
5.846.3 Member Function Documentation . . . 2818
5.847 std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > Class Template Reference . . . 2819
5.847.1 Detailed Description . . . 2820
5.847.2 Constructor & Destructor Documentation . . . 2820
5.847.3 Member Function Documentation . . . 2823

5.848 std::regex_traits< _Ch_type > Class Template Reference . . . 2824
5.848.1 Detailed Description . . . 2824
5.848.2 Constructor & Destructor Documentation . . . 2825
5.848.3 Member Function Documentation . . . 2825
5.849 std::remove_all_extents< _Tp > Struct Template Reference . . . 2829
5.849.1 Detailed Description . . . 2830
5.850 std::remove_const< _Tp > Struct Template Reference . . . 2830
5.850.1 Detailed Description . . . 2830
5.851 std::remove_cv< _Tp > Struct Template Reference . . . 2830
5.851.1 Detailed Description . . . 2830
5.852 std::remove_extent< _Tp > Struct Template Reference . . . 2830
5.852.1 Detailed Description . . . 2831
5.853 std::remove_pointer< _Tp > Struct Template Reference . . . 2831
5.853.1 Detailed Description . . . 2831
5.854 std::remove_reference< _Tp > Struct Template Reference . . . 2831
5.854.1 Detailed Description . . . 2831
5.855 std::remove_volatile< _Tp > Struct Template Reference . . . 2831
5.855.1 Detailed Description . . . 2832
5.856 __gnu_pbds::resize_error Struct Reference . . . 2832
5.856.1 Detailed Description . . . 2832
5.856.2 Member Function Documentation . . . 2832
5.857 __gnu_pbds::detail::resize_policy< _Tp > Class Template Reference . . . 2833
5.857.1 Detailed Description . . . 2833
5.858 std::result_of< _Signature > Struct Template Reference . . . 2833
5.858.1 Detailed Description . . . 2833
5.859 std::reverse_iterator< _Iterator > Class Template Reference . . . 2834
5.859.1 Detailed Description . . . 2835
5.859.2 Constructor & Destructor Documentation . . . 2835
5.859.3 Member Function Documentation . . . 2836
5.860 __gnu_cxx::rope< _CharT, _Alloc > Class Template Reference . . . 2838
5.860.1 Detailed Description . . . 2843
5.861 std::runtime_error Class Reference . . . 2843
5.861.1 Detailed Description . . . 2844
5.861.2 Constructor & Destructor Documentation . . . 2844
5.861.3 Member Function Documentation . . . 2844
5.862 __gnu_pbds::sample_probe_fn Class Reference . . . 2844
5.862.1 Detailed Description . . . 2844
5.862.2 Constructor & Destructor Documentation . . . 2845
5.862.3 Member Function Documentation . . . 2845

5.863 __gnu_pbds::sample_range_hashing Class Reference . . . 2845
5.863.1 Detailed Description . . . 2845
5.863.2 Member Typedef Documentation . . . 2846
5.863.3 Constructor & Destructor Documentation . . . 2846
5.863.4 Member Function Documentation . . . 2846
5.864 __gnu_pbds::sample_ranged_hash_fn Class Reference . . . 2846
5.864.1 Detailed Description . . . 2847
5.864.2 Constructor & Destructor Documentation . . . 2847
5.864.3 Member Function Documentation . . . 2847
5.865 __gnu_pbds::sample_ranged_probe_fn Class Reference . . . 2847
5.865.1 Detailed Description . . . 2848
5.866 __gnu_pbds::sample_resize_policy Class Reference . . . 2848
5.866.1 Detailed Description . . . 2849
5.866.2 Member Typedef Documentation . . . 2849
5.866.3 Constructor & Destructor Documentation . . . 2849
5.866.4 Member Function Documentation . . . 2849
5.867 __gnu_pbds::sample_resize_trigger Class Reference . . . 2851
5.867.1 Detailed Description . . . 2851
5.867.2 Member Typedef Documentation . . . 2851
5.867.3 Constructor & Destructor Documentation . . . 2851
5.867.4 Member Function Documentation . . . 2852
5.868 __gnu_pbds::sample_size_policy Class Reference . . . 2853
5.868.1 Detailed Description . . . 2854
5.868.2 Member Typedef Documentation . . . 2854
5.868.3 Constructor & Destructor Documentation . . . 2854
5.868.4 Member Function Documentation . . . 2854
5.869 __gnu_pbds::sample_tree_node_update< Const_Node_Iter, Node_Iter, Cmp_Fn, _Alloc > Class Template Reference . . . 2855
5.869.1 Detailed Description . . . 2855
5.870 __gnu_pbds::sample_trie_access_traits Struct Reference . . . 2855
5.870.1 Detailed Description . . . 2855
5.870.2 Member Typedef Documentation . . . 2855
5.870.3 Member Function Documentation . . . 2855
5.871 __gnu_pbds::sample_trie_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc > Class Template Reference . . . 2856
5.871.1 Detailed Description . . . 2856
5.871.2 Constructor & Destructor Documentation . . . 2856
5.871.3 Member Function Documentation . . . 2856
5.872 __gnu_pbds::sample_update_policy Struct Reference . . . 2857
5.872.1 Detailed Description . . . 2857

5.872.2 Member Typedef Documentation . . . 2857
5.872.3 Constructor & Destructor Documentation . . . 2857
5.872.4 Member Function Documentation . . . 2857
5.873 __gnu_parallel::sampling_tag Struct Reference . . . 2858
5.873.1 Detailed Description . . . 2858
5.873.2 Member Function Documentation . . . 2858
5.874 std::scoped_allocator_adaptor< _OuterAlloc, _InnerAllocs > Class Template Reference . . . 2859
5.874.1 Detailed Description . . . 2860
5.875 std::seed_seq Class Reference . . . 2860
5.875.1 Detailed Description . . . 2861
5.875.2 Member Typedef Documentation . . . 2861
5.875.3 Constructor & Destructor Documentation . . . 2861
5.876 __gnu_cxx::select1st< _Pair > Struct Template Reference . . . 2861
5.876.1 Detailed Description . . . 2862
5.876.2 Member Typedef Documentation . . . 2862
5.877 __gnu_cxx::select2nd< _Pair > Struct Template Reference . . . 2862
5.877.1 Detailed Description . . . 2862
5.877.2 Member Typedef Documentation . . . 2862
5.878 __gnu_pbds::detail::select_value_type< Key, Mapped > Struct Template Reference . . . 2863
5.878.1 Detailed Description . . . 2863
5.879 __gnu_pbds::detail::select_value_type< Key, null_type > Struct Template Reference . . . 2863
5.879.1 Detailed Description . . . 2863
5.880 std::basic_istream< _CharT, _Traits >::sentry Class Reference . . . 2863
5.880.1 Detailed Description . . . 2869
5.880.2 Member Typedef Documentation . . . 2869
5.880.3 Member Enumeration Documentation . . . 2870
5.880.4 Constructor & Destructor Documentation . . . 2870
5.880.5 Member Function Documentation . . . 2871
5.880.6 Member Data Documentation . . . 2894
5.881 std::basic_ostream< _CharT, _Traits >::sentry Class Reference . . . 2898
5.881.1 Detailed Description . . . 2902
5.881.2 Member Typedef Documentation . . . 2902
5.881.3 Member Enumeration Documentation . . . 2903
5.881.4 Constructor & Destructor Documentation . . . 2904
5.881.5 Member Function Documentation . . . 2904
5.881.6 Member Data Documentation . . . 2922
5.882 __gnu_pbds::sequence_tag Struct Reference . . . 2925
5.882.1 Detailed Description . . . 2926
5.883 __gnu_parallel::sequential_tag Struct Reference . . . 2926

5.883.1 Detailed Description . . . 2926
5.884 std::__debug::set< _Key, _Compare, _Allocator > Class Template Reference . . . 2926
5.884.1 Detailed Description . . . 2929
5.885 std::set< _Key, _Compare, _Alloc > Class Template Reference . . . 2929
5.885.1 Detailed Description . . . 2932
5.885.2 Member Typedef Documentation . . . 2932
5.885.3 Constructor & Destructor Documentation . . . 2934
5.885.4 Member Function Documentation . . . 2937
5.886 std::shared_future< _Res > Class Template Reference . . . 2952
5.886.1 Detailed Description . . . 2953
5.886.2 Constructor & Destructor Documentation . . . 2953
5.886.3 Member Function Documentation . . . 2953
5.887 std::shared_future< _Res & > Class Template Reference . . . 2954
5.887.1 Detailed Description . . . 2955
5.887.2 Constructor & Destructor Documentation . . . 2955
5.887.3 Member Function Documentation . . . 2955
5.888 std::shared_future< void > Class Reference . . . 2956
5.888.1 Detailed Description . . . 2957
5.888.2 Constructor & Destructor Documentation . . . 2957
5.888.3 Member Function Documentation . . . 2958
5.889 std::shared_lock< _Mutex > Class Template Reference . . . 2958
5.889.1 Detailed Description . . . 2959
5.890 std::shared_ptr< _Tp > Class Template Reference . . . 2959
5.890.1 Detailed Description . . . 2961
5.890.2 Member Typedef Documentation . . . 2961
5.890.3 Constructor & Destructor Documentation . . . 2961
5.890.4 Member Function Documentation . . . 2967
5.891 std::shared_timed_mutex Class Reference . . . 2968
5.891.1 Detailed Description . . . 2969
5.892 std::shuffle_order_engine< _RandomNumberEngine, __k > Class Template Reference . . . 2969
5.892.1 Detailed Description . . . 2970
5.892.2 Member Typedef Documentation . . . 2970
5.892.3 Constructor & Destructor Documentation . . . 2970
5.892.4 Member Function Documentation . . . 2971
5.892.5 Friends And Related Symbol Documentation . . . 2972
5.893 std::slice Class Reference . . . 2974
5.893.1 Detailed Description . . . 2974
5.893.2 Friends And Related Symbol Documentation . . . 2974
5.894 std::slice_array< _Tp > Class Template Reference . . . 2974

5.894.1 Detailed Description . . . 2975
5.894.2 Member Function Documentation . . . 2976
5.895 __gnu_cxx::slist< _Tp, _Alloc > Class Template Reference . . . 2977
5.895.1 Detailed Description . . . 2979
5.896 std::experimental::filesystem::v1::space_info Struct Reference . . . 2979
5.896.1 Detailed Description . . . 2979
5.897 std::filesystem::space_info Struct Reference . . . 2979
5.897.1 Detailed Description . . . 2980
5.898 __gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > Class Template Reference . . . 2980
5.898.1 Detailed Description . . . 2982
5.898.2 Member Function Documentation . . . 2983
5.899 __gnu_pbds::detail::splay_tree_node_< Value_Type, Metadata, _Alloc > Struct Template Reference . . . 2983
5.899.1 Detailed Description . . . 2984
5.900 __gnu_pbds::splay_tree_tag Struct Reference . . . 2984
5.900.1 Detailed Description . . . 2985
5.901 std::stack< _Tp, _Sequence > Class Template Reference . . . 2985
5.901.1 Detailed Description . . . 2986
5.901.2 Constructor & Destructor Documentation . . . 2987
5.901.3 Member Function Documentation . . . 2987
5.902 __gnu_cxx::stdio_filebuf< _CharT, _Traits > Class Template Reference . . . 2988
5.902.1 Detailed Description . . . 2991
5.902.2 Constructor & Destructor Documentation . . . 2991
5.902.3 Member Function Documentation . . . 2992
5.902.4 Member Data Documentation . . . 3007
5.903 __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits > Class Template Reference . . . 3010
5.903.1 Detailed Description . . . 3012
5.903.2 Member Function Documentation . . . 3012
5.903.3 Member Data Documentation . . . 3025
5.904 std::chrono::steady_clock Struct Reference . . . 3026
5.904.1 Detailed Description . . . 3027
5.905 __gnu_pbds::detail::stored_data< _Tv, _Th, Store_Hash > Struct Template Reference . . . 3027
5.905.1 Detailed Description . . . 3027
5.906 __gnu_pbds::detail::stored_data< _Tv, _Th, false > Struct Template Reference . . . 3028
5.906.1 Detailed Description . . . 3028
5.907 __gnu_pbds::detail::stored_hash< _Th > Struct Template Reference . . . 3028
5.907.1 Detailed Description . . . 3029
5.908 __gnu_pbds::detail::stored_value< _Tv > Struct Template Reference . . . 3029
5.908.1 Detailed Description . . . 3030

5.909 __gnu_pbds::string_tag Struct Reference . . . 3030
5.909.1 Detailed Description . . . 3031
5.910 std::student_t_distribution< _RealType > Class Template Reference . . . 3031
5.910.1 Detailed Description . . . 3032
5.910.2 Member Typedef Documentation . . . 3032
5.910.3 Member Function Documentation . . . 3032
5.910.4 Friends And Related Symbol Documentation . . . 3033
5.911 std::sub_match< _BiIter > Class Template Reference . . . 3034
5.911.1 Detailed Description . . . 3035
5.911.2 Member Function Documentation . . . 3036
5.912 std::ranges::subrange< _It, _Sent, _Kind > Class Template Reference . . . 3037
5.912.1 Detailed Description . . . 3039
5.913 std::subtract_with_carry_engine< _UIntType, __w, __s, __r > Class Template Reference . . . 3039
5.913.1 Detailed Description . . . 3040
5.913.2 Member Typedef Documentation . . . 3040
5.913.3 Constructor & Destructor Documentation . . . 3040
5.913.4 Member Function Documentation . . . 3041
5.913.5 Friends And Related Symbol Documentation . . . 3042
5.914 __gnu_cxx::subtractive_rng Class Reference . . . 3043
5.914.1 Detailed Description . . . 3044
5.914.2 Member Typedef Documentation . . . 3044
5.914.3 Constructor & Destructor Documentation . . . 3044
5.914.4 Member Function Documentation . . . 3044
5.915 __gnu_pbds::detail::synth_access_traits< Type_Traits, Set, _ATraits > Struct Template Reference . . . 3044
5.915.1 Detailed Description . . . 3045
5.916 std::chrono::system_clock Struct Reference . . . 3045
5.916.1 Detailed Description . . . 3045
5.917 std::system_error Class Reference . . . 3046
5.917.1 Detailed Description . . . 3046
5.917.2 Member Function Documentation . . . 3046
5.918 std::chrono::tai_clock Class Reference . . . 3046
5.918.1 Detailed Description . . . 3047
5.919 __gnu_cxx::temporary_buffer< _ForwardIterator, _Tp > Struct Template Reference . . . 3047
5.919.1 Detailed Description . . . 3048
5.919.2 Constructor & Destructor Documentation . . . 3048
5.919.3 Member Function Documentation . . . 3049
5.920 __gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc > Class Template Reference . . . 3049
5.920.1 Detailed Description . . . 3051
5.921 __gnu_pbds::thin_heap_tag Struct Reference . . . 3051

5.921.1 Detailed Description . . . 3052
5.922 std::thread Class Reference . . . 3052
5.922.1 Detailed Description . . . 3052
5.922.2 Member Function Documentation . . . 3053
5.923 __gnu_cxx::throw_allocator_base< _Tp, _Cond > Class Template Reference . . . 3053
5.923.1 Detailed Description . . . 3054
5.924 __gnu_cxx::throw_allocator_limit< _Tp > Struct Template Reference . . . 3054
5.924.1 Detailed Description . . . 3056
5.925 __gnu_cxx::throw_allocator_random< _Tp > Struct Template Reference . . . 3056
5.925.1 Detailed Description . . . 3058
5.926 __gnu_cxx::throw_value_base< _Cond > Struct Template Reference . . . 3058
5.926.1 Detailed Description . . . 3058
5.927 __gnu_cxx::throw_value_limit Struct Reference . . . 3059
5.927.1 Detailed Description . . . 3060
5.928 __gnu_cxx::throw_value_random Struct Reference . . . 3060
5.928.1 Detailed Description . . . 3061
5.929 std::time_base Class Reference . . . 3061
5.929.1 Detailed Description . . . 3062
5.930 std::time_get< _CharT, _InIter > Class Template Reference . . . 3062
5.930.1 Detailed Description . . . 3063
5.930.2 Member Typedef Documentation . . . 3064
5.930.3 Constructor & Destructor Documentation . . . 3064
5.930.4 Member Function Documentation . . . 3064
5.930.5 Member Data Documentation . . . 3073
5.931 std::time_get_byname< _CharT, _InIter > Class Template Reference . . . 3073
5.931.1 Detailed Description . . . 3075
5.931.2 Member Function Documentation . . . 3075
5.931.3 Member Data Documentation . . . 3083
5.932 std::chrono::time_point< _Clock, _Dur > Class Template Reference . . . 3083
5.932.1 Detailed Description . . . 3084
5.933 std::time_put< _CharT, _OutIter > Class Template Reference . . . 3084
5.933.1 Detailed Description . . . 3086
5.933.2 Member Typedef Documentation . . . 3086
5.933.3 Constructor & Destructor Documentation . . . 3086
5.933.4 Member Function Documentation . . . 3086
5.933.5 Member Data Documentation . . . 3088
5.934 std::time_put_byname< _CharT, _OutIter > Class Template Reference . . . 3088
5.934.1 Detailed Description . . . 3090
5.934.2 Member Function Documentation . . . 3090

5.934.3 Member Data Documentation . . . 3091
5.935 std::timed_mutex Class Reference . . . 3092
5.935.1 Detailed Description . . . 3092
5.936 std::to_chars_result Struct Reference . . . 3092
5.936.1 Detailed Description . . . 3092
5.937 std::chrono::treat_as_floating_point< _Rep > Struct Template Reference . . . 3093
5.937.1 Detailed Description . . . 3093
5.938 __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc > Class Template Reference . . . 3093
5.938.1 Detailed Description . . . 3094
5.938.2 Member Typedef Documentation . . . 3095
5.938.3 Constructor & Destructor Documentation . . . 3095
5.939 __gnu_pbds::detail::tree_metadata_helper< Node_Update, _BTp > Struct Template Reference . . . 3096
5.939.1 Detailed Description . . . 3096
5.940 __gnu_pbds::detail::tree_metadata_helper< Node_Update, false > Struct Template Reference . . . 3096
5.940.1 Detailed Description . . . 3096
5.941 __gnu_pbds::detail::tree_metadata_helper< Node_Update, true > Struct Template Reference . . . 3096
5.941.1 Detailed Description . . . 3096
5.942 __gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc > Struct Template Reference . . . 3097
5.942.1 Detailed Description . . . 3097
5.943 __gnu_pbds::tree_order_statistics_node_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc > Class Template Reference . . . 3097
5.943.1 Detailed Description . . . 3098
5.943.2 Member Function Documentation . . . 3098
5.944 __gnu_pbds::tree_tag Struct Reference . . . 3099
5.944.1 Detailed Description . . . 3099
5.945 __gnu_pbds::detail::tree_traits< Key, Data, Cmp_Fn, Node_Update, Tag, _Alloc > Struct Template Reference . . . 3099
5.945.1 Detailed Description . . . 3099
5.946 __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc > Struct Template Reference . . . 3100
5.946.1 Detailed Description . . . 3100
5.946.2 Member Typedef Documentation . . . 3100
5.947 __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc > Struct Template Reference . . . 3100
5.947.1 Detailed Description . . . 3101
5.947.2 Member Typedef Documentation . . . 3101
5.948 __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc > Struct Template Reference . . . 3101
5.948.1 Detailed Description . . . 3102
5.948.2 Member Typedef Documentation . . . 3103

5.949 __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc > Struct Template Reference . . . 3103
5.949.1 Detailed Description . . . 3103
5.949.2 Member Typedef Documentation . . . 3103
5.950 __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc > Struct Template Reference . . . 3103
5.950.1 Detailed Description . . . 3104
5.950.2 Member Typedef Documentation . . . 3105
5.951 __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc > Struct Template Reference . . . 3105
5.951.1 Detailed Description . . . 3106
5.951.2 Member Typedef Documentation . . . 3106
5.952 __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc > Class Template Reference . . . 3106
5.952.1 Detailed Description . . . 3107
5.952.2 Member Typedef Documentation . . . 3108
5.952.3 Constructor & Destructor Documentation . . . 3108
5.953 __gnu_pbds::detail::trie_metadata_helper< Node_Update, _BTp > Struct Template Reference . . . 3109
5.953.1 Detailed Description . . . 3109
5.954 __gnu_pbds::detail::trie_metadata_helper< Node_Update, false > Struct Template Reference . . . 3109
5.954.1 Detailed Description . . . 3109
5.955 __gnu_pbds::detail::trie_metadata_helper< Node_Update, true > Struct Template Reference . . . 3109
5.955.1 Detailed Description . . . 3110
5.956 __gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc > Struct Template Reference . . . 3110
5.956.1 Detailed Description . . . 3110
5.957 __gnu_pbds::trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc > Class Template Reference . . . 3110
5.957.1 Detailed Description . . . 3112
5.957.2 Member Function Documentation . . . 3112
5.958 __gnu_pbds::detail::trie_policy_base< Node_CItr, Node_Itr, _ATraits, _Alloc > Class Template Reference . . . 3113
5.958.1 Detailed Description . . . 3114
5.958.2 Member Function Documentation . . . 3114
5.959 __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc > Class Template Reference . . . 3114
5.959.1 Detailed Description . . . 3116
5.959.2 Member Typedef Documentation . . . 3116
5.959.3 Member Function Documentation . . . 3116
5.960 __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc > Struct Template Reference . . . 3117
5.960.1 Detailed Description . . . 3118
5.960.2 Member Typedef Documentation . . . 3118

5.960.3 Member Function Documentation . . . 3118
5.961 __gnu_pbds::trie_tag Struct Reference . . . 3119
5.961.1 Detailed Description . . . 3120
5.962 __gnu_pbds::detail::trie_traits< Key, Data, _ATraits, Node_Update, Tag, _Alloc > Struct Template Reference . . . 3120
5.962.1 Detailed Description . . . 3120
5.963 __gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc > Struct Template Reference . . . 3121
5.963.1 Detailed Description . . . 3121
5.963.2 Member Typedef Documentation . . . 3121
5.964 __gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc > Struct Template Reference . . . 3122
5.964.1 Detailed Description . . . 3122
5.964.2 Member Typedef Documentation . . . 3122
5.965 __gnu_pbds::trivial_iterator_tag Struct Reference . . . 3123
5.965.1 Detailed Description . . . 3123
5.966 std::try_to_lock_t Struct Reference . . . 3123
5.966.1 Detailed Description . . . 3123
5.967 std::tuple< _Elements > Class Template Reference . . . 3123
5.967.1 Detailed Description . . . 3125
5.968 std::tuple< _T1, _T2 > Class Template Reference . . . 3125
5.968.1 Detailed Description . . . 3127
5.969 std::tuple_element< __i, _Tp > Struct Template Reference . . . 3127
5.969.1 Detailed Description . . . 3127
5.970 std::tuple_element< 0, pair< _Tp1, _Tp2 > > Struct Template Reference . . . 3128
5.970.1 Detailed Description . . . 3128
5.971 std::tuple_element< 1, pair< _Tp1, _Tp2 > > Struct Template Reference . . . 3128
5.971.1 Detailed Description . . . 3128
5.972 std::tuple_element< __i, tuple< _Types... > > Struct Template Reference . . . 3128
5.972.1 Detailed Description . . . 3128
5.973 std::tuple_element< _Ind, array< _Tp, _Nm > > Struct Template Reference . . . 3128
5.973.1 Detailed Description . . . 3129
5.974 std::tuple_size< _Tp > Struct Template Reference . . . 3129
5.974.1 Detailed Description . . . 3129
5.975 std::tuple_size< array< _Tp, _Nm > > Struct Template Reference . . . 3129
5.975.1 Detailed Description . . . 3130
5.976 std::tuple_size< pair< _Tp1, _Tp2 > > Struct Template Reference . . . 3130
5.976.1 Detailed Description . . . 3131
5.977 std::tuple_size< tuple< _Elements... > > Struct Template Reference . . . 3131
5.977.1 Detailed Description . . . 3131

5.978 __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, false >::type Struct Reference 3132
5.978.1 Detailed Description 3132
5.979 std::type_index Struct Reference 3132
5.979.1 Detailed Description 3132
5.980 __gnu_debug::type_info Class Reference 3132
5.980.1 Detailed Description 3133
5.980.2 Constructor & Destructor Documentation 3133
5.980.3 Member Function Documentation 3133
5.981 std::type_info Class Reference 3133
5.981.1 Detailed Description 3134
5.981.2 Constructor & Destructor Documentation 3134
5.981.3 Member Function Documentation 3134
5.982 __gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash > Struct Template Reference . . 3134
5.982.1 Detailed Description 3136
5.983 std::chrono::tzdb_list Class Reference 3136
5.983.1 Detailed Description 3136
5.983.2 Member Function Documentation 3137
5.984 __gnu_cxx::unary_compose< _Operation1, _Operation2 > Class Template Reference 3137
5.984.1 Detailed Description 3138
5.984.2 Member Typedef Documentation 3138
5.985 std::unary_function< _Arg, _Result > Struct Template Reference 3139
5.985.1 Detailed Description 3139
5.985.2 Member Typedef Documentation 3139
5.986 std::unary_negate< _Predicate > Class Template Reference 3140
5.986.1 Detailed Description 3140
5.986.2 Member Typedef Documentation 3140
5.987 __gnu_parallel::unbalanced_tag Struct Reference 3141
5.987.1 Detailed Description 3141
5.987.2 Member Function Documentation 3141
5.988 std::underflow_error Class Reference 3142
5.988.1 Detailed Description 3142
5.988.2 Member Function Documentation 3142
5.989 std::underlying_type< _Tp > Struct Template Reference 3143
5.989.1 Detailed Description 3143
5.990 std::uniform_int_distribution< _IntType > Class Template Reference 3143
5.990.1 Detailed Description 3144
5.990.2 Member Typedef Documentation 3144
5.990.3 Constructor & Destructor Documentation 3144
5.990.4 Member Function Documentation 3144

5.990.5 Friends And Related Symbol Documentation . . . 3145
5.991 std::uniform_real_distribution< _RealType > Class Template Reference . . . 3145
5.991.1 Detailed Description . . . 3146
5.991.2 Member Typedef Documentation . . . 3146
5.991.3 Constructor & Destructor Documentation . . . 3146
5.991.4 Member Function Documentation . . . 3147
5.991.5 Friends And Related Symbol Documentation . . . 3148
5.992 std::unique_lock< _Mutex > Class Template Reference . . . 3148
5.992.1 Detailed Description . . . 3149
5.992.2 Friends And Related Symbol Documentation . . . 3149
5.993 std::unique_ptr< _Tp, _Dp > Class Template Reference . . . 3149
5.993.1 Detailed Description . . . 3150
5.993.2 Constructor & Destructor Documentation . . . 3151
5.993.3 Member Function Documentation . . . 3152
5.994 std::unique_ptr< _Tp[], _Dp > Class Template Reference . . . 3154
5.994.1 Detailed Description . . . 3156
5.994.2 Constructor & Destructor Documentation . . . 3157
5.994.3 Member Function Documentation . . . 3160
5.994.4 Friends And Related Symbol Documentation . . . 3163
5.995 std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference . . . 3164
5.995.1 Detailed Description . . . 3167
5.996 std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference . . . 3167
5.996.1 Detailed Description . . . 3170
5.996.2 Member Typedef Documentation . . . 3171
5.996.3 Constructor & Destructor Documentation . . . 3173
5.996.4 Member Function Documentation . . . 3175
5.997 std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference . 3195
5.997.1 Detailed Description . . . 3198
5.998 std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference . . . 3198
5.998.1 Detailed Description . . . 3201
5.998.2 Member Typedef Documentation . . . 3201
5.998.3 Constructor & Destructor Documentation . . . 3204
5.998.4 Member Function Documentation . . . 3206
5.999 std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc > Class Template Reference . . . 3224
5.999.1 Detailed Description . . . 3226
5.1000 std::unordered_multiset< _Value, _Hash, _Pred, _Alloc > Class Template Reference . . . 3226
5.1000.1 Detailed Description . . . 3229
5.1000.2 Member Typedef Documentation . . . 3230
5.1000.3 Constructor & Destructor Documentation . . . 3232

5.1000.4 Member Function Documentation 3234
5.1001 std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc > Class Template Reference 3250
5.1001.1 Detailed Description 3253
5.1002 std::unordered_set< _Value, _Hash, _Pred, _Alloc > Class Template Reference 3253
5.1002.1 Detailed Description 3256
5.1002.2 Member Typedef Documentation 3256
5.1002.3 Constructor & Destructor Documentation 3258
5.1002.4 Member Function Documentation 3260
5.1003 std::pmr::unsynchronized_pool_resource Class Reference 3278
5.1003.1 Detailed Description 3279
5.1003.2 Member Function Documentation 3279
5.1004 std::uses_allocator< typename, typename > Struct Template Reference 3280
5.1004.1 Detailed Description 3280
5.1005 std::uses_allocator< tuple< _Types... >, _Alloc > Struct Template Reference 3280
5.1005.1 Detailed Description 3280
5.1006 std::chrono::utc_clock Class Reference 3280
5.1006.1 Detailed Description 3280
5.1007 std::valarray< _Tp > Class Template Reference 3281
5.1007.1 Detailed Description 3283
5.1007.2 Constructor & Destructor Documentation 3283
5.1007.3 Member Function Documentation 3283
5.1008 std::__debug::vector< _Tp, _Allocator > Class Template Reference 3286
5.1008.1 Detailed Description 3288
5.1008.2 Constructor & Destructor Documentation 3289
5.1009 std::vector< _Tp, _Alloc > Class Template Reference 3289
5.1009.1 Detailed Description 3292
5.1009.2 Constructor & Destructor Documentation 3293
5.1009.3 Member Function Documentation 3295
5.1010 std::vector< bool, _Alloc > Class Template Reference 3308
5.1010.1 Detailed Description 3313
5.1010.2 Constructor & Destructor Documentation 3313
5.1010.3 Member Function Documentation 3316
5.1011 std::ranges::view_interface< _Derived > Class Template Reference 3328
5.1011.1 Detailed Description 3328
5.1012 std::wbuffer_convert< _Codecvt, _Elem, _Tr > Class Template Reference 3329
5.1012.1 Detailed Description 3331
5.1012.2 Member Typedef Documentation 3331
5.1012.3 Constructor & Destructor Documentation 3331
5.1012.4 Member Function Documentation 3332

5.1012.5 Member Data Documentation . . . 3344
5.1013 std::weak_ptr< _Tp > Class Template Reference . . . 3345
5.1013.1 Detailed Description . . . 3346
5.1014 std::weibull_distribution< _RealType > Class Template Reference . . . 3346
5.1014.1 Detailed Description . . . 3347
5.1014.2 Member Typedef Documentation . . . 3347
5.1014.3 Member Function Documentation . . . 3347
5.1014.4 Friends And Related Symbol Documentation . . . 3348
5.1015 std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc > Class Template Reference . . . 3348
5.1015.1 Detailed Description . . . 3349
5.1015.2 Constructor & Destructor Documentation . . . 3349
5.1015.3 Member Function Documentation . . . 3350

6 File Documentation **3352**
6.1 algorithm File Reference . . . 3352
6.1.1 Detailed Description . . . 3352
6.2 experimental/algorithm File Reference . . . 3352
6.2.1 Detailed Description . . . 3353
6.3 ext/algorithm File Reference . . . 3353
6.3.1 Detailed Description . . . 3354
6.4 parallel/algorithm File Reference . . . 3354
6.4.1 Detailed Description . . . 3354
6.5 any File Reference . . . 3354
6.5.1 Detailed Description . . . 3354
6.6 experimental/any File Reference . . . 3355
6.6.1 Detailed Description . . . 3355
6.7 array File Reference . . . 3355
6.7.1 Detailed Description . . . 3356
6.8 experimental/array File Reference . . . 3356
6.8.1 Detailed Description . . . 3357
6.9 atomic File Reference . . . 3357
6.9.1 Detailed Description . . . 3360
6.10 auto_ptr.h File Reference . . . 3361
6.10.1 Detailed Description . . . 3361
6.11 backward_warning.h File Reference . . . 3361
6.11.1 Detailed Description . . . 3361
6.12 hash_fun.h File Reference . . . 3361
6.12.1 Detailed Description . . . 3361
6.13 hash_map File Reference . . . 3361

6.13.1 Detailed Description . . . 3362
6.14 hash_set File Reference . . . 3362
6.14.1 Detailed Description . . . 3363
6.15 strstream File Reference . . . 3363
6.15.1 Detailed Description . . . 3363
6.16 barrier File Reference . . . 3363
6.16.1 Detailed Description . . . 3363
6.17 bit File Reference . . . 3363
6.17.1 Detailed Description . . . 3364
6.18 bits/algorithmfwd.h File Reference . . . 3364
6.18.1 Detailed Description . . . 3369
6.19 parallel/algorithmfwd.h File Reference . . . 3369
6.19.1 Detailed Description . . . 3377
6.20 align.h File Reference . . . 3377
6.20.1 Detailed Description . . . 3377
6.21 bits/alloc_traits.h File Reference . . . 3377
6.21.1 Detailed Description . . . 3378
6.22 ext/alloc_traits.h File Reference . . . 3378
6.22.1 Detailed Description . . . 3378
6.23 allocated_ptr.h File Reference . . . 3378
6.23.1 Detailed Description . . . 3378
6.24 allocator.h File Reference . . . 3378
6.24.1 Detailed Description . . . 3378
6.25 atomic_base.h File Reference . . . 3378
6.25.1 Detailed Description . . . 3379
6.26 atomic_futex.h File Reference . . . 3379
6.26.1 Detailed Description . . . 3379
6.27 atomic_lockfree_defines.h File Reference . . . 3379
6.27.1 Detailed Description . . . 3380
6.28 atomic_timed_wait.h File Reference . . . 3380
6.28.1 Detailed Description . . . 3380
6.29 atomic_wait.h File Reference . . . 3380
6.29.1 Detailed Description . . . 3380
6.30 basic_ios.h File Reference . . . 3380
6.30.1 Detailed Description . . . 3380
6.31 basic_ios.tcc File Reference . . . 3380
6.31.1 Detailed Description . . . 3381
6.32 basic_string.h File Reference . . . 3381
6.32.1 Detailed Description . . . 3383

6.33 basic_string.tcc File Reference . . . 3383
6.33.1 Detailed Description . . . 3384
6.34 backward/binders.h File Reference . . . 3384
6.34.1 Detailed Description . . . 3384
6.35 boost_concept_check.h File Reference . . . 3384
6.35.1 Detailed Description . . . 3385
6.36 c++0x_warning.h File Reference . . . 3385
6.36.1 Detailed Description . . . 3385
6.37 char_traits.h File Reference . . . 3385
6.37.1 Detailed Description . . . 3385
6.38 charconv.h File Reference . . . 3386
6.38.1 Detailed Description . . . 3386
6.39 chrono.h File Reference . . . 3386
6.39.1 Detailed Description . . . 3389
6.39.2 Function Documentation . . . 3390
6.40 chrono_io.h File Reference . . . 3390
6.40.1 Detailed Description . . . 3393
6.41 codecvt.h File Reference . . . 3394
6.41.1 Detailed Description . . . 3394
6.42 concept_check.h File Reference . . . 3394
6.42.1 Detailed Description . . . 3394
6.43 cow_string.h File Reference . . . 3394
6.43.1 Detailed Description . . . 3395
6.44 cpp_type_traits.h File Reference . . . 3395
6.44.1 Detailed Description . . . 3395
6.45 cpyfunc_impl.h File Reference . . . 3395
6.45.1 Detailed Description . . . 3396
6.46 cxxabi_forced.h File Reference . . . 3396
6.46.1 Detailed Description . . . 3396
6.47 cxxabi_init_exception.h File Reference . . . 3396
6.47.1 Detailed Description . . . 3396
6.48 deque.tcc File Reference . . . 3396
6.48.1 Detailed Description . . . 3398
6.49 enable_special_members.h File Reference . . . 3398
6.49.1 Detailed Description . . . 3398
6.50 erase_if.h File Reference . . . 3398
6.50.1 Detailed Description . . . 3398
6.51 exception.h File Reference . . . 3398
6.51.1 Detailed Description . . . 3399

6.52 exception_defines.h File Reference . . . 3399
6.52.1 Detailed Description . . . 3399
6.53 exception_ptr.h File Reference . . . 3399
6.53.1 Detailed Description . . . 3399
6.53.2 Function Documentation . . . 3399
6.54 formatfwd.h File Reference . . . 3399
6.54.1 Detailed Description . . . 3399
6.55 forward_list.h File Reference . . . 3400
6.55.1 Detailed Description . . . 3400
6.56 forward_list.tcc File Reference . . . 3400
6.56.1 Detailed Description . . . 3401
6.57 bits/fs_dir.h File Reference . . . 3401
6.57.1 Detailed Description . . . 3401
6.58 experimental/bits/fs_dir.h File Reference . . . 3401
6.58.1 Detailed Description . . . 3402
6.59 bits/fs_fwd.h File Reference . . . 3402
6.59.1 Detailed Description . . . 3404
6.60 experimental/bits/fs_fwd.h File Reference . . . 3404
6.60.1 Detailed Description . . . 3405
6.61 bits/fs_ops.h File Reference . . . 3405
6.61.1 Detailed Description . . . 3407
6.62 experimental/bits/fs_ops.h File Reference . . . 3408
6.62.1 Detailed Description . . . 3410
6.63 bits/fs_path.h File Reference . . . 3410
6.63.1 Detailed Description . . . 3410
6.64 experimental/bits/fs_path.h File Reference . . . 3410
6.64.1 Detailed Description . . . 3411
6.65 fstream.tcc File Reference . . . 3411
6.65.1 Detailed Description . . . 3411
6.66 funcref_impl.h File Reference . . . 3411
6.66.1 Detailed Description . . . 3411
6.67 functexcept.h File Reference . . . 3411
6.67.1 Detailed Description . . . 3412
6.68 functional_hash.h File Reference . . . 3412
6.68.1 Detailed Description . . . 3413
6.69 funcwrap.h File Reference . . . 3413
6.69.1 Detailed Description . . . 3413
6.70 gslice.h File Reference . . . 3413
6.70.1 Detailed Description . . . 3413

6.71 gslice_array.h File Reference . . . 3413
6.71.1 Detailed Description . . . 3413
6.72 hash_bytes.h File Reference . . . 3413
6.72.1 Detailed Description . . . 3414
6.73 backward/hashtable.h File Reference . . . 3414
6.73.1 Detailed Description . . . 3414
6.74 bits/hashtable.h File Reference . . . 3414
6.74.1 Detailed Description . . . 3414
6.75 hashtable_policy.h File Reference . . . 3415
6.75.1 Detailed Description . . . 3415
6.76 indirect.h File Reference . . . 3415
6.76.1 Detailed Description . . . 3415
6.77 indirect_array.h File Reference . . . 3415
6.77.1 Detailed Description . . . 3415
6.78 invoke.h File Reference . . . 3415
6.78.1 Detailed Description . . . 3416
6.79 ios_base.h File Reference . . . 3416
6.79.1 Detailed Description . . . 3417
6.80 istream.tcc File Reference . . . 3417
6.80.1 Detailed Description . . . 3418
6.81 iterator_concepts.h File Reference . . . 3418
6.81.1 Detailed Description . . . 3418
6.82 list.tcc File Reference . . . 3418
6.82.1 Detailed Description . . . 3418
6.83 locale_classes.h File Reference . . . 3419
6.83.1 Detailed Description . . . 3419
6.84 locale_classes.tcc File Reference . . . 3419
6.84.1 Detailed Description . . . 3419
6.85 locale_conv.h File Reference . . . 3419
6.85.1 Detailed Description . . . 3420
6.86 locale_facets.h File Reference . . . 3420
6.86.1 Detailed Description . . . 3422
6.87 locale_facets.tcc File Reference . . . 3422
6.87.1 Detailed Description . . . 3422
6.88 locale_facets_nonio.h File Reference . . . 3422
6.88.1 Detailed Description . . . 3423
6.89 locale_facets_nonio.tcc File Reference . . . 3423
6.89.1 Detailed Description . . . 3423
6.90 localefwd.h File Reference . . . 3424

6.90.1 Detailed Description . . . 3424
6.91 mask_array.h File Reference . . . 3424
6.91.1 Detailed Description . . . 3425
6.92 max_size_type.h File Reference . . . 3425
6.92.1 Detailed Description . . . 3425
6.93 memory_resource.h File Reference . . . 3425
6.93.1 Detailed Description . . . 3425
6.94 memoryfwd.h File Reference . . . 3425
6.94.1 Detailed Description . . . 3425
6.95 mofunc_impl.h File Reference . . . 3425
6.95.1 Detailed Description . . . 3426
6.96 monostate.h File Reference . . . 3426
6.96.1 Detailed Description . . . 3426
6.97 move.h File Reference . . . 3426
6.97.1 Detailed Description . . . 3427
6.98 nested_exception.h File Reference . . . 3427
6.98.1 Detailed Description . . . 3427
6.99 bits/new_allocator.h File Reference . . . 3427
6.99.1 Detailed Description . . . 3428
6.100 ext/new_allocator.h File Reference . . . 3428
6.100.1 Detailed Description . . . 3428
6.101 node_handle.h File Reference . . . 3428
6.101.1 Detailed Description . . . 3428
6.102 ostream.h File Reference . . . 3428
6.102.1 Detailed Description . . . 3429
6.103 ostream.tcc File Reference . . . 3430
6.103.1 Detailed Description . . . 3430
6.104 ostream_insert.h File Reference . . . 3430
6.104.1 Detailed Description . . . 3430
6.105 out_ptr.h File Reference . . . 3430
6.105.1 Detailed Description . . . 3430
6.106 parse_numbers.h File Reference . . . 3430
6.106.1 Detailed Description . . . 3430
6.107 postypes.h File Reference . . . 3431
6.107.1 Detailed Description . . . 3431
6.108 predefined_ops.h File Reference . . . 3431
6.108.1 Detailed Description . . . 3432
6.109 ptr_traits.h File Reference . . . 3432
6.109.1 Detailed Description . . . 3432

6.110 quoted_string.h File Reference . . . 3432
6.110.1 Detailed Description . . . 3433
6.111 random.h File Reference . . . 3433
6.111.1 Detailed Description . . . 3435
6.112 bits/random.tcc File Reference . . . 3435
6.112.1 Detailed Description . . . 3439
6.113 ext/random.tcc File Reference . . . 3439
6.113.1 Detailed Description . . . 3441
6.114 range_access.h File Reference . . . 3441
6.114.1 Detailed Description . . . 3443
6.115 ranges_algo.h File Reference . . . 3443
6.115.1 Detailed Description . . . 3447
6.116 ranges_algobase.h File Reference . . . 3447
6.116.1 Detailed Description . . . 3447
6.117 ranges_base.h File Reference . . . 3447
6.117.1 Detailed Description . . . 3447
6.118 ranges_cmp.h File Reference . . . 3448
6.118.1 Detailed Description . . . 3448
6.119 ranges_uninitialized.h File Reference . . . 3448
6.119.1 Detailed Description . . . 3448
6.120 ranges_util.h File Reference . . . 3448
6.120.1 Detailed Description . . . 3449
6.121 refwrap.h File Reference . . . 3449
6.121.1 Detailed Description . . . 3449
6.122 regex.h File Reference . . . 3450
6.122.1 Detailed Description . . . 3452
6.123 regex.tcc File Reference . . . 3452
6.123.1 Detailed Description . . . 3452
6.124 regex_automaton.h File Reference . . . 3452
6.124.1 Detailed Description . . . 3453
6.125 regex_automaton.tcc File Reference . . . 3453
6.125.1 Detailed Description . . . 3453
6.126 regex_compiler.h File Reference . . . 3453
6.126.1 Detailed Description . . . 3453
6.127 regex_compiler.tcc File Reference . . . 3454
6.127.1 Detailed Description . . . 3454
6.128 regex_constants.h File Reference . . . 3454
6.128.1 Detailed Description . . . 3455
6.129 regex_error.h File Reference . . . 3455

6.129.1 Detailed Description . . . 3456
6.130 regex_executor.h File Reference . . . 3456
6.130.1 Detailed Description . . . 3456
6.131 regex_executor.tcc File Reference . . . 3456
6.131.1 Detailed Description . . . 3456
6.132 regex_scanner.h File Reference . . . 3457
6.132.1 Detailed Description . . . 3457
6.133 regex_scanner.tcc File Reference . . . 3457
6.133.1 Detailed Description . . . 3457
6.134 requires_hosted.h File Reference . . . 3457
6.134.1 Detailed Description . . . 3457
6.135 sat_arith.h File Reference . . . 3457
6.135.1 Detailed Description . . . 3457
6.136 semaphore_base.h File Reference . . . 3457
6.136.1 Detailed Description . . . 3457
6.137 bits/shared_ptr.h File Reference . . . 3457
6.137.1 Detailed Description . . . 3458
6.138 experimental/bits/shared_ptr.h File Reference . . . 3458
6.138.1 Detailed Description . . . 3460
6.139 shared_ptr_atomic.h File Reference . . . 3460
6.139.1 Detailed Description . . . 3462
6.140 shared_ptr_base.h File Reference . . . 3462
6.140.1 Detailed Description . . . 3463
6.141 slice_array.h File Reference . . . 3463
6.141.1 Detailed Description . . . 3463
6.142 specfun.h File Reference . . . 3463
6.142.1 Detailed Description . . . 3465
6.143 sstream.tcc File Reference . . . 3466
6.143.1 Detailed Description . . . 3466
6.144 std_abs.h File Reference . . . 3466
6.144.1 Detailed Description . . . 3466
6.145 std_function.h File Reference . . . 3466
6.145.1 Detailed Description . . . 3467
6.146 std_mutex.h File Reference . . . 3467
6.146.1 Detailed Description . . . 3467
6.147 std_thread.h File Reference . . . 3467
6.147.1 Detailed Description . . . 3468
6.148 stl_algo.h File Reference . . . 3468
6.148.1 Detailed Description . . . 3477

6.149 stl_algobase.h File Reference . . . 3477
6.149.1 Detailed Description . . . 3482
6.150 stl_bvector.h File Reference . . . 3482
6.150.1 Detailed Description . . . 3483
6.151 stl_construct.h File Reference . . . 3483
6.151.1 Detailed Description . . . 3484
6.152 stl_deque.h File Reference . . . 3484
6.152.1 Detailed Description . . . 3484
6.152.2 Macro Definition Documentation . . . 3484
6.153 stl_function.h File Reference . . . 3485
6.153.1 Detailed Description . . . 3486
6.154 stl_heap.h File Reference . . . 3486
6.154.1 Detailed Description . . . 3488
6.155 bits/stl_iterator.h File Reference . . . 3488
6.155.1 Detailed Description . . . 3490
6.156 debug/stl_iterator.h File Reference . . . 3490
6.156.1 Detailed Description . . . 3491
6.157 stl_iterator_base_funcs.h File Reference . . . 3491
6.157.1 Detailed Description . . . 3492
6.158 stl_iterator_base_types.h File Reference . . . 3492
6.158.1 Detailed Description . . . 3493
6.159 stl_list.h File Reference . . . 3493
6.159.1 Detailed Description . . . 3494
6.160 stl_map.h File Reference . . . 3494
6.160.1 Detailed Description . . . 3494
6.161 stl_multimap.h File Reference . . . 3495
6.161.1 Detailed Description . . . 3495
6.162 stl_multiset.h File Reference . . . 3496
6.162.1 Detailed Description . . . 3496
6.163 stl_numeric.h File Reference . . . 3497
6.163.1 Detailed Description . . . 3497
6.164 stl_pair.h File Reference . . . 3497
6.164.1 Detailed Description . . . 3498
6.165 stl_queue.h File Reference . . . 3498
6.165.1 Detailed Description . . . 3499
6.166 stl_raw_storage_iter.h File Reference . . . 3500
6.166.1 Detailed Description . . . 3500
6.167 stl_relops.h File Reference . . . 3500
6.167.1 Detailed Description . . . 3500

6.168 stl_set.h File Reference . . . 3500
6.168.1 Detailed Description . . . 3501
6.169 stl_stack.h File Reference . . . 3501
6.169.1 Detailed Description . . . 3502
6.170 stl_tempbuf.h File Reference . . . 3502
6.170.1 Detailed Description . . . 3502
6.171 stl_tree.h File Reference . . . 3503
6.171.1 Detailed Description . . . 3503
6.172 stl_uninitialized.h File Reference . . . 3503
6.172.1 Detailed Description . . . 3504
6.173 stl_vector.h File Reference . . . 3504
6.173.1 Detailed Description . . . 3504
6.174 stream_iterator.h File Reference . . . 3505
6.174.1 Detailed Description . . . 3505
6.175 streambuf.tcc File Reference . . . 3505
6.175.1 Detailed Description . . . 3505
6.176 streambuf_iterator.h File Reference . . . 3505
6.176.1 Detailed Description . . . 3506
6.177 bits/string_view.tcc File Reference . . . 3506
6.177.1 Detailed Description . . . 3506
6.178 experimental/bits/string_view.tcc File Reference . . . 3506
6.178.1 Detailed Description . . . 3507
6.179 stringfwd.h File Reference . . . 3507
6.179.1 Detailed Description . . . 3507
6.180 text_encoding-data.h File Reference . . . 3507
6.180.1 Detailed Description . . . 3507
6.181 this_thread_sleep.h File Reference . . . 3507
6.181.1 Detailed Description . . . 3507
6.182 unicode-data.h File Reference . . . 3508
6.182.1 Detailed Description . . . 3508
6.183 unicode.h File Reference . . . 3508
6.183.1 Detailed Description . . . 3509
6.184 uniform_int_dist.h File Reference . . . 3509
6.184.1 Detailed Description . . . 3510
6.185 unique_lock.h File Reference . . . 3510
6.185.1 Detailed Description . . . 3510
6.186 unique_ptr.h File Reference . . . 3510
6.186.1 Detailed Description . . . 3511
6.187 unordered_map.h File Reference . . . 3511

6.187.1 Detailed Description . . . 3513
6.188 unordered_set.h File Reference . . . 3513
6.188.1 Detailed Description . . . 3515
6.189 uses_allocator_args.h File Reference . . . 3516
6.189.1 Detailed Description . . . 3516
6.190 utility.h File Reference . . . 3516
6.190.1 Detailed Description . . . 3517
6.191 valarray_after.h File Reference . . . 3517
6.191.1 Detailed Description . . . 3527
6.192 valarray_array.h File Reference . . . 3527
6.192.1 Detailed Description . . . 3534
6.193 valarray_array.tcc File Reference . . . 3535
6.193.1 Detailed Description . . . 3535
6.194 valarray_before.h File Reference . . . 3535
6.194.1 Detailed Description . . . 3535
6.195 vector.tcc File Reference . . . 3536
6.195.1 Detailed Description . . . 3536
6.196 version.h File Reference . . . 3536
6.196.1 Detailed Description . . . 3538
6.197 bitset File Reference . . . 3539
6.197.1 Detailed Description . . . 3539
6.198 debug/bitset File Reference . . . 3539
6.198.1 Detailed Description . . . 3540
6.199 cassert File Reference . . . 3540
6.199.1 Detailed Description . . . 3540
6.200 ccomplex File Reference . . . 3540
6.200.1 Detailed Description . . . 3540
6.201 tr1/ccomplex File Reference . . . 3540
6.201.1 Detailed Description . . . 3541
6.202 cctype File Reference . . . 3541
6.202.1 Detailed Description . . . 3541
6.203 tr1/cctype File Reference . . . 3541
6.203.1 Detailed Description . . . 3541
6.204 cerrno File Reference . . . 3541
6.204.1 Detailed Description . . . 3541
6.205 cfenv File Reference . . . 3541
6.205.1 Detailed Description . . . 3541
6.206 tr1/cfenv File Reference . . . 3542
6.206.1 Detailed Description . . . 3542

6.207 cfloat File Reference . . . 3542
6.207.1 Detailed Description . . . 3542
6.208 tr1/cfloat File Reference . . . 3542
6.208.1 Detailed Description . . . 3542
6.209 charconv File Reference . . . 3542
6.209.1 Detailed Description . . . 3544
6.210 chrono File Reference . . . 3544
6.210.1 Detailed Description . . . 3547
6.211 experimental/chrono File Reference . . . 3547
6.211.1 Detailed Description . . . 3547
6.212 cinttypes File Reference . . . 3547
6.212.1 Detailed Description . . . 3547
6.213 tr1/cinttypes File Reference . . . 3547
6.213.1 Detailed Description . . . 3547
6.214 ciso646 File Reference . . . 3547
6.214.1 Detailed Description . . . 3547
6.215 climits File Reference . . . 3548
6.215.1 Detailed Description . . . 3548
6.216 tr1/climits File Reference . . . 3548
6.216.1 Detailed Description . . . 3548
6.217 clocale File Reference . . . 3548
6.217.1 Detailed Description . . . 3548
6.218 cmath File Reference . . . 3548
6.218.1 Detailed Description . . . 3559
6.219 ext/cmath File Reference . . . 3559
6.219.1 Detailed Description . . . 3560
6.220 tr1/cmath File Reference . . . 3560
6.220.1 Detailed Description . . . 3562
6.221 codecvt File Reference . . . 3562
6.221.1 Detailed Description . . . 3562
6.222 complex File Reference . . . 3563
6.222.1 Detailed Description . . . 3567
6.223 tr1/complex File Reference . . . 3567
6.223.1 Detailed Description . . . 3568
6.224 complex.h File Reference . . . 3568
6.224.1 Detailed Description . . . 3568
6.225 concepts File Reference . . . 3568
6.225.1 Detailed Description . . . 3568
6.226 condition_variable File Reference . . . 3568

6.226.1 Detailed Description . . . 3569
6.227 csetjmp File Reference . . . 3569
6.227.1 Detailed Description . . . 3569
6.228 csignal File Reference . . . 3569
6.228.1 Detailed Description . . . 3569
6.229 cstdalign File Reference . . . 3570
6.229.1 Detailed Description . . . 3570
6.230 cstdarg File Reference . . . 3570
6.230.1 Detailed Description . . . 3570
6.231 tr1/cstdarg File Reference . . . 3570
6.231.1 Detailed Description . . . 3570
6.232 cstdbool File Reference . . . 3570
6.232.1 Detailed Description . . . 3570
6.233 tr1/cstdbool File Reference . . . 3571
6.233.1 Detailed Description . . . 3571
6.234 cstddef File Reference . . . 3571
6.234.1 Detailed Description . . . 3571
6.235 cstdint File Reference . . . 3571
6.235.1 Detailed Description . . . 3571
6.236 tr1/cstdint File Reference . . . 3571
6.236.1 Detailed Description . . . 3571
6.237 cstdio File Reference . . . 3572
6.237.1 Detailed Description . . . 3572
6.238 tr1/cstdio File Reference . . . 3572
6.238.1 Detailed Description . . . 3572
6.239 cstdlib File Reference . . . 3572
6.239.1 Detailed Description . . . 3572
6.240 tr1/cstdlib File Reference . . . 3572
6.240.1 Detailed Description . . . 3573
6.241 cstring File Reference . . . 3573
6.241.1 Detailed Description . . . 3573
6.242 ctgmath File Reference . . . 3573
6.242.1 Detailed Description . . . 3573
6.243 tr1/ctgmath File Reference . . . 3573
6.243.1 Detailed Description . . . 3573
6.244 ctime File Reference . . . 3574
6.244.1 Detailed Description . . . 3574
6.245 tr1/ctime File Reference . . . 3574
6.245.1 Detailed Description . . . 3574

6.246 cuchar File Reference . . . 3574
6.246.1 Detailed Description . . . 3574
6.247 cwchar File Reference . . . 3574
6.247.1 Detailed Description . . . 3575
6.248 tr1/cwchar File Reference . . . 3575
6.248.1 Detailed Description . . . 3575
6.249 cwctype File Reference . . . 3575
6.249.1 Detailed Description . . . 3575
6.250 tr1/cwctype File Reference . . . 3575
6.250.1 Detailed Description . . . 3575
6.251 assertions.h File Reference . . . 3576
6.251.1 Detailed Description . . . 3576
6.252 debug.h File Reference . . . 3576
6.252.1 Detailed Description . . . 3576
6.253 formatter.h File Reference . . . 3577
6.253.1 Detailed Description . . . 3577
6.254 functions.h File Reference . . . 3577
6.254.1 Detailed Description . . . 3579
6.255 helper_functions.h File Reference . . . 3579
6.255.1 Detailed Description . . . 3581
6.256 macros.h File Reference . . . 3581
6.256.1 Detailed Description . . . 3582
6.256.2 Macro Definition Documentation . . . 3582
6.257 map.h File Reference . . . 3584
6.257.1 Detailed Description . . . 3585
6.258 multimap.h File Reference . . . 3585
6.258.1 Detailed Description . . . 3585
6.259 multiset.h File Reference . . . 3586
6.259.1 Detailed Description . . . 3586
6.260 safe_base.h File Reference . . . 3586
6.260.1 Detailed Description . . . 3587
6.261 safe_container.h File Reference . . . 3587
6.261.1 Detailed Description . . . 3587
6.262 safe_iterator.h File Reference . . . 3587
6.262.1 Detailed Description . . . 3588
6.263 safe_iterator.tcc File Reference . . . 3588
6.263.1 Detailed Description . . . 3589
6.264 safe_local_iterator.h File Reference . . . 3589
6.264.1 Detailed Description . . . 3589

6.265 safe_local_iterator.tcc File Reference . . . 3590
6.265.1 Detailed Description . . . 3590
6.266 safe_sequence.h File Reference . . . 3590
6.266.1 Detailed Description . . . 3590
6.267 safe_sequence.tcc File Reference . . . 3590
6.267.1 Detailed Description . . . 3590
6.268 safe_unordered_base.h File Reference . . . 3590
6.268.1 Detailed Description . . . 3591
6.269 safe_unordered_container.h File Reference . . . 3591
6.269.1 Detailed Description . . . 3591
6.270 safe_unordered_container.tcc File Reference . . . 3591
6.270.1 Detailed Description . . . 3591
6.271 set.h File Reference . . . 3591
6.271.1 Detailed Description . . . 3592
6.272 decimal File Reference . . . 3592
6.272.1 Detailed Description . . . 3601
6.273 debug/deque File Reference . . . 3601
6.273.1 Detailed Description . . . 3602
6.274 deque File Reference . . . 3602
6.274.1 Detailed Description . . . 3602
6.275 experimental/deque File Reference . . . 3602
6.275.1 Detailed Description . . . 3603
6.276 expected File Reference . . . 3603
6.276.1 Detailed Description . . . 3603
6.277 lfts_config.h File Reference . . . 3603
6.277.1 Detailed Description . . . 3603
6.278 ext/numeric_traits.h File Reference . . . 3603
6.278.1 Detailed Description . . . 3604
6.279 propagate_const File Reference . . . 3604
6.279.1 Detailed Description . . . 3605
6.280 simd File Reference . . . 3605
6.280.1 Detailed Description . . . 3605
6.281 aligned_buffer.h File Reference . . . 3605
6.281.1 Detailed Description . . . 3605
6.282 atomicity.h File Reference . . . 3606
6.282.1 Detailed Description . . . 3606
6.283 bitmap_allocator.h File Reference . . . 3606
6.283.1 Detailed Description . . . 3607
6.283.2 Macro Definition Documentation . . . 3607

6.284 cast.h File Reference . . . 3607
6.284.1 Detailed Description . . . 3608
6.285 codecvt_specializations.h File Reference . . . 3608
6.285.1 Detailed Description . . . 3608
6.286 concurrence.h File Reference . . . 3608
6.286.1 Detailed Description . . . 3609
6.287 debug_allocator.h File Reference . . . 3609
6.287.1 Detailed Description . . . 3609
6.288 enc_filebuf.h File Reference . . . 3609
6.288.1 Detailed Description . . . 3609
6.289 extptr_allocator.h File Reference . . . 3609
6.289.1 Detailed Description . . . 3610
6.290 malloc_allocator.h File Reference . . . 3610
6.290.1 Detailed Description . . . 3610
6.291 mt_allocator.h File Reference . . . 3610
6.291.1 Detailed Description . . . 3610
6.292 assoc_container.hpp File Reference . . . 3611
6.292.1 Detailed Description . . . 3611
6.293 bin_search_tree_.hpp File Reference . . . 3611
6.293.1 Detailed Description . . . 3611
6.294 bin_search_tree_/node_iterators.hpp File Reference . . . 3612
6.294.1 Detailed Description . . . 3612
6.295 ov_tree_map_/node_iterators.hpp File Reference . . . 3612
6.295.1 Detailed Description . . . 3612
6.296 point_iterators.hpp File Reference . . . 3612
6.296.1 Detailed Description . . . 3613
6.297 bin_search_tree_/r_erase_fn_imps.hpp File Reference . . . 3613
6.297.1 Detailed Description . . . 3613
6.298 pat_trie_/r_erase_fn_imps.hpp File Reference . . . 3613
6.298.1 Detailed Description . . . 3613
6.299 bin_search_tree_/rotate_fn_imps.hpp File Reference . . . 3613
6.299.1 Detailed Description . . . 3613
6.300 pat_trie_/rotate_fn_imps.hpp File Reference . . . 3613
6.300.1 Detailed Description . . . 3613
6.301 bin_search_tree_/traits.hpp File Reference . . . 3613
6.301.1 Detailed Description . . . 3613
6.302 branch_policy/traits.hpp File Reference . . . 3613
6.302.1 Detailed Description . . . 3613
6.303 ov_tree_map_/traits.hpp File Reference . . . 3614

6.303.1 Detailed Description . . . 3614
6.304 pat_trie_/traits.hpp File Reference . . . 3614
6.304.1 Detailed Description . . . 3614
6.305 rb_tree_map_/traits.hpp File Reference . . . 3614
6.305.1 Detailed Description . . . 3614
6.306 splay_tree_/traits.hpp File Reference . . . 3614
6.306.1 Detailed Description . . . 3615
6.307 binary_heap_.hpp File Reference . . . 3615
6.307.1 Detailed Description . . . 3615
6.308 binary_heap_/const_iterator.hpp File Reference . . . 3615
6.308.1 Detailed Description . . . 3615
6.309 left_child_next_sibling_heap_/const_iterator.hpp File Reference . . . 3616
6.309.1 Detailed Description . . . 3616
6.310 unordered_iterator/const_iterator.hpp File Reference . . . 3616
6.310.1 Detailed Description . . . 3616
6.311 bin_search_tree_/constructors_destructor_fn_imps.hpp File Reference . . . 3616
6.311.1 Detailed Description . . . 3616
6.312 binary_heap_/constructors_destructor_fn_imps.hpp File Reference . . . 3616
6.312.1 Detailed Description . . . 3616
6.313 binomial_heap_/constructors_destructor_fn_imps.hpp File Reference . . . 3616
6.313.1 Detailed Description . . . 3616
6.314 binomial_heap_base_/constructors_destructor_fn_imps.hpp File Reference . . . 3616
6.314.1 Detailed Description . . . 3616
6.315 left_child_next_sibling_heap_/constructors_destructor_fn_imps.hpp File Reference . . . 3616
6.315.1 Detailed Description . . . 3616
6.316 ov_tree_map_/constructors_destructor_fn_imps.hpp File Reference . . . 3617
6.316.1 Detailed Description . . . 3617
6.317 pairing_heap_/constructors_destructor_fn_imps.hpp File Reference . . . 3617
6.317.1 Detailed Description . . . 3617
6.318 pat_trie_/constructors_destructor_fn_imps.hpp File Reference . . . 3617
6.318.1 Detailed Description . . . 3617
6.319 rb_tree_map_/constructors_destructor_fn_imps.hpp File Reference . . . 3617
6.319.1 Detailed Description . . . 3617
6.320 rc_binomial_heap_/constructors_destructor_fn_imps.hpp File Reference . . . 3617
6.320.1 Detailed Description . . . 3617
6.321 splay_tree_/constructors_destructor_fn_imps.hpp File Reference . . . 3617
6.321.1 Detailed Description . . . 3617
6.322 thin_heap_/constructors_destructor_fn_imps.hpp File Reference . . . 3617
6.322.1 Detailed Description . . . 3617

6.323 bin_search_tree_/debug_fn_imps.hpp File Reference . . . 3617
6.323.1 Detailed Description . . . 3617
6.324 binary_heap_/debug_fn_imps.hpp File Reference . . . 3617
6.324.1 Detailed Description . . . 3617
6.325 binomial_heap_/debug_fn_imps.hpp File Reference . . . 3617
6.325.1 Detailed Description . . . 3617
6.326 binomial_heap_base_/debug_fn_imps.hpp File Reference . . . 3618
6.326.1 Detailed Description . . . 3618
6.327 cc_hash_table_map_/debug_fn_imps.hpp File Reference . . . 3618
6.327.1 Detailed Description . . . 3618
6.328 gp_hash_table_map_/debug_fn_imps.hpp File Reference . . . 3618
6.328.1 Detailed Description . . . 3618
6.329 left_child_next_sibling_heap_/debug_fn_imps.hpp File Reference . . . 3618
6.329.1 Detailed Description . . . 3618
6.330 list_update_map_/debug_fn_imps.hpp File Reference . . . 3618
6.330.1 Detailed Description . . . 3618
6.331 ov_tree_map_/debug_fn_imps.hpp File Reference . . . 3618
6.331.1 Detailed Description . . . 3618
6.332 pairing_heap_/debug_fn_imps.hpp File Reference . . . 3618
6.332.1 Detailed Description . . . 3618
6.333 pat_trie_/debug_fn_imps.hpp File Reference . . . 3618
6.333.1 Detailed Description . . . 3618
6.334 rb_tree_map_/debug_fn_imps.hpp File Reference . . . 3618
6.334.1 Detailed Description . . . 3618
6.335 rc_binomial_heap_/debug_fn_imps.hpp File Reference . . . 3618
6.335.1 Detailed Description . . . 3618
6.336 splay_tree_/debug_fn_imps.hpp File Reference . . . 3619
6.336.1 Detailed Description . . . 3619
6.337 thin_heap_/debug_fn_imps.hpp File Reference . . . 3619
6.337.1 Detailed Description . . . 3619
6.338 entry_cmp.hpp File Reference . . . 3619
6.338.1 Detailed Description . . . 3619
6.339 entry_pred.hpp File Reference . . . 3619
6.339.1 Detailed Description . . . 3619
6.340 bin_search_tree_/erase_fn_imps.hpp File Reference . . . 3619
6.340.1 Detailed Description . . . 3619
6.341 binary_heap_/erase_fn_imps.hpp File Reference . . . 3619
6.341.1 Detailed Description . . . 3619
6.342 binomial_heap_base_/erase_fn_imps.hpp File Reference . . . 3620

6.342.1 Detailed Description . . . 3620
6.343 cc_hash_table_map_/erase_fn_imps.hpp File Reference . . . 3620
6.343.1 Detailed Description . . . 3620
6.344 gp_hash_table_map_/erase_fn_imps.hpp File Reference . . . 3620
6.344.1 Detailed Description . . . 3620
6.345 left_child_next_sibling_heap_/erase_fn_imps.hpp File Reference . . . 3620
6.345.1 Detailed Description . . . 3620
6.346 list_update_map_/erase_fn_imps.hpp File Reference . . . 3620
6.346.1 Detailed Description . . . 3620
6.347 ov_tree_map_/erase_fn_imps.hpp File Reference . . . 3620
6.347.1 Detailed Description . . . 3620
6.348 pairing_heap_/erase_fn_imps.hpp File Reference . . . 3620
6.348.1 Detailed Description . . . 3620
6.349 pat_trie_/erase_fn_imps.hpp File Reference . . . 3620
6.349.1 Detailed Description . . . 3620
6.350 rb_tree_map_/erase_fn_imps.hpp File Reference . . . 3620
6.350.1 Detailed Description . . . 3620
6.351 rc_binomial_heap_/erase_fn_imps.hpp File Reference . . . 3620
6.351.1 Detailed Description . . . 3620
6.352 splay_tree_/erase_fn_imps.hpp File Reference . . . 3621
6.352.1 Detailed Description . . . 3621
6.353 thin_heap_/erase_fn_imps.hpp File Reference . . . 3621
6.353.1 Detailed Description . . . 3621
6.354 bin_search_tree_/find_fn_imps.hpp File Reference . . . 3621
6.354.1 Detailed Description . . . 3621
6.355 binary_heap_/find_fn_imps.hpp File Reference . . . 3621
6.355.1 Detailed Description . . . 3621
6.356 binomial_heap_base_/find_fn_imps.hpp File Reference . . . 3621
6.356.1 Detailed Description . . . 3621
6.357 cc_hash_table_map_/find_fn_imps.hpp File Reference . . . 3621
6.357.1 Detailed Description . . . 3621
6.358 gp_hash_table_map_/find_fn_imps.hpp File Reference . . . 3621
6.358.1 Detailed Description . . . 3621
6.359 list_update_map_/find_fn_imps.hpp File Reference . . . 3621
6.359.1 Detailed Description . . . 3621
6.360 pairing_heap_/find_fn_imps.hpp File Reference . . . 3621
6.360.1 Detailed Description . . . 3621
6.361 pat_trie_/find_fn_imps.hpp File Reference . . . 3621
6.361.1 Detailed Description . . . 3621

6.362 rb_tree_map_/find_fn_imps.hpp File Reference . . . 3622
6.362.1 Detailed Description . . . 3622
6.363 splay_tree_/find_fn_imps.hpp File Reference . . . 3622
6.363.1 Detailed Description . . . 3622
6.364 thin_heap_/find_fn_imps.hpp File Reference . . . 3622
6.364.1 Detailed Description . . . 3622
6.365 bin_search_tree_/info_fn_imps.hpp File Reference . . . 3622
6.365.1 Detailed Description . . . 3622
6.366 binary_heap_/info_fn_imps.hpp File Reference . . . 3622
6.366.1 Detailed Description . . . 3622
6.367 cc_hash_table_map_/info_fn_imps.hpp File Reference . . . 3622
6.367.1 Detailed Description . . . 3622
6.368 gp_hash_table_map_/info_fn_imps.hpp File Reference . . . 3622
6.368.1 Detailed Description . . . 3622
6.369 left_child_next_sibling_heap_/info_fn_imps.hpp File Reference . . . 3622
6.369.1 Detailed Description . . . 3622
6.370 list_update_map_/info_fn_imps.hpp File Reference . . . 3622
6.370.1 Detailed Description . . . 3622
6.371 ov_tree_map_/info_fn_imps.hpp File Reference . . . 3622
6.371.1 Detailed Description . . . 3622
6.372 pat_trie_/info_fn_imps.hpp File Reference . . . 3623
6.372.1 Detailed Description . . . 3623
6.373 rb_tree_map_/info_fn_imps.hpp File Reference . . . 3623
6.373.1 Detailed Description . . . 3623
6.374 splay_tree_/info_fn_imps.hpp File Reference . . . 3623
6.374.1 Detailed Description . . . 3623
6.375 bin_search_tree_/insert_fn_imps.hpp File Reference . . . 3623
6.375.1 Detailed Description . . . 3623
6.376 binary_heap_/insert_fn_imps.hpp File Reference . . . 3623
6.376.1 Detailed Description . . . 3623
6.377 binomial_heap_base_/insert_fn_imps.hpp File Reference . . . 3623
6.377.1 Detailed Description . . . 3623
6.378 cc_hash_table_map_/insert_fn_imps.hpp File Reference . . . 3623
6.378.1 Detailed Description . . . 3623
6.379 gp_hash_table_map_/insert_fn_imps.hpp File Reference . . . 3623
6.379.1 Detailed Description . . . 3623
6.380 left_child_next_sibling_heap_/insert_fn_imps.hpp File Reference . . . 3623
6.380.1 Detailed Description . . . 3623
6.381 list_update_map_/insert_fn_imps.hpp File Reference . . . 3623

6.381.1 Detailed Description . . . 3623
6.382 ov_tree_map_/insert_fn_imps.hpp File Reference . . . 3624
6.382.1 Detailed Description . . . 3624
6.383 pairing_heap_/insert_fn_imps.hpp File Reference . . . 3624
6.383.1 Detailed Description . . . 3624
6.384 rb_tree_map_/insert_fn_imps.hpp File Reference . . . 3624
6.384.1 Detailed Description . . . 3624
6.385 rc_binomial_heap_/insert_fn_imps.hpp File Reference . . . 3624
6.385.1 Detailed Description . . . 3624
6.386 splay_tree_/insert_fn_imps.hpp File Reference . . . 3624
6.386.1 Detailed Description . . . 3624
6.387 thin_heap_/insert_fn_imps.hpp File Reference . . . 3624
6.387.1 Detailed Description . . . 3624
6.388 bin_search_tree_/iterators_fn_imps.hpp File Reference . . . 3624
6.388.1 Detailed Description . . . 3624
6.389 binary_heap_/iterators_fn_imps.hpp File Reference . . . 3624
6.389.1 Detailed Description . . . 3624
6.390 cc_hash_table_map_/iterators_fn_imps.hpp File Reference . . . 3624
6.390.1 Detailed Description . . . 3624
6.391 left_child_next_sibling_heap_/iterators_fn_imps.hpp File Reference . . . 3624
6.391.1 Detailed Description . . . 3624
6.392 list_update_map_/iterators_fn_imps.hpp File Reference . . . 3625
6.392.1 Detailed Description . . . 3625
6.393 ov_tree_map_/iterators_fn_imps.hpp File Reference . . . 3625
6.393.1 Detailed Description . . . 3625
6.394 pat_trie_/iterators_fn_imps.hpp File Reference . . . 3625
6.394.1 Detailed Description . . . 3625
6.395 binary_heap_/point_const_iterator.hpp File Reference . . . 3625
6.395.1 Detailed Description . . . 3625
6.396 left_child_next_sibling_heap_/point_const_iterator.hpp File Reference . . . 3625
6.396.1 Detailed Description . . . 3625
6.397 unordered_iterator/point_const_iterator.hpp File Reference . . . 3625
6.397.1 Detailed Description . . . 3625
6.398 bin_search_tree_/policy_access_fn_imps.hpp File Reference . . . 3626
6.398.1 Detailed Description . . . 3626
6.399 binary_heap_/policy_access_fn_imps.hpp File Reference . . . 3626
6.399.1 Detailed Description . . . 3626
6.400 cc_hash_table_map_/policy_access_fn_imps.hpp File Reference . . . 3626
6.400.1 Detailed Description . . . 3626

6.401 gp_hash_table_map_/policy_access_fn_imps.hpp File Reference . . . 3626
6.401.1 Detailed Description . . . 3626
6.402 left_child_next_sibling_heap_/policy_access_fn_imps.hpp File Reference . . . 3626
6.402.1 Detailed Description . . . 3626
6.403 ov_tree_map_/policy_access_fn_imps.hpp File Reference . . . 3626
6.403.1 Detailed Description . . . 3626
6.404 pat_trie_/policy_access_fn_imps.hpp File Reference . . . 3626
6.404.1 Detailed Description . . . 3626
6.405 resize_policy.hpp File Reference . . . 3626
6.405.1 Detailed Description . . . 3627
6.406 bin_search_tree_/split_join_fn_imps.hpp File Reference . . . 3627
6.406.1 Detailed Description . . . 3627
6.407 binary_heap_/split_join_fn_imps.hpp File Reference . . . 3627
6.407.1 Detailed Description . . . 3627
6.408 binomial_heap_base_/split_join_fn_imps.hpp File Reference . . . 3627
6.408.1 Detailed Description . . . 3627
6.409 ov_tree_map_/split_join_fn_imps.hpp File Reference . . . 3627
6.409.1 Detailed Description . . . 3627
6.410 pairing_heap_/split_join_fn_imps.hpp File Reference . . . 3627
6.410.1 Detailed Description . . . 3627
6.411 rb_tree_map_/split_join_fn_imps.hpp File Reference . . . 3627
6.411.1 Detailed Description . . . 3627
6.412 rc_binomial_heap_/split_join_fn_imps.hpp File Reference . . . 3627
6.412.1 Detailed Description . . . 3627
6.413 splay_tree_/split_join_fn_imps.hpp File Reference . . . 3627
6.413.1 Detailed Description . . . 3627
6.414 thin_heap_/split_join_fn_imps.hpp File Reference . . . 3627
6.414.1 Detailed Description . . . 3627
6.415 binary_heap_/trace_fn_imps.hpp File Reference . . . 3628
6.415.1 Detailed Description . . . 3628
6.416 cc_hash_table_map_/trace_fn_imps.hpp File Reference . . . 3628
6.416.1 Detailed Description . . . 3628
6.417 gp_hash_table_map_/trace_fn_imps.hpp File Reference . . . 3628
6.417.1 Detailed Description . . . 3628
6.418 left_child_next_sibling_heap_/trace_fn_imps.hpp File Reference . . . 3628
6.418.1 Detailed Description . . . 3628
6.419 list_update_map_/trace_fn_imps.hpp File Reference . . . 3628
6.419.1 Detailed Description . . . 3628
6.420 pat_trie_/trace_fn_imps.hpp File Reference . . . 3628

6.420.1 Detailed Description . . . 3628
6.421 rc_binomial_heap_/trace_fn_imps.hpp File Reference . . . 3628
6.421.1 Detailed Description . . . 3628
6.422 thin_heap_/trace_fn_imps.hpp File Reference . . . 3628
6.422.1 Detailed Description . . . 3628
6.423 binomial_heap_.hpp File Reference . . . 3628
6.423.1 Detailed Description . . . 3629
6.424 binomial_heap_base_.hpp File Reference . . . 3629
6.424.1 Detailed Description . . . 3629
6.425 branch_policy.hpp File Reference . . . 3629
6.425.1 Detailed Description . . . 3629
6.426 null_node_metadata.hpp File Reference . . . 3629
6.426.1 Detailed Description . . . 3630
6.427 cc_ht_map_.hpp File Reference . . . 3630
6.427.1 Detailed Description . . . 3630
6.428 cmp_fn_imps.hpp File Reference . . . 3630
6.428.1 Detailed Description . . . 3630
6.429 cond_key_dtor_entry_dealtor.hpp File Reference . . . 3631
6.429.1 Detailed Description . . . 3631
6.430 cc_hash_table_map_/constructor_destructor_fn_imps.hpp File Reference . . . 3631
6.430.1 Detailed Description . . . 3631
6.431 gp_hash_table_map_/constructor_destructor_fn_imps.hpp File Reference . . . 3631
6.431.1 Detailed Description . . . 3631
6.432 list_update_map_/constructor_destructor_fn_imps.hpp File Reference . . . 3631
6.433 cc_hash_table_map_/constructor_destructor_no_store_hash_fn_imps.hpp File Reference . . . 3631
6.433.1 Detailed Description . . . 3631
6.434 gp_hash_table_map_/constructor_destructor_no_store_hash_fn_imps.hpp File Reference . . . 3631
6.434.1 Detailed Description . . . 3631
6.435 cc_hash_table_map_/constructor_destructor_store_hash_fn_imps.hpp File Reference . . . 3631
6.435.1 Detailed Description . . . 3631
6.436 gp_hash_table_map_/constructor_destructor_store_hash_fn_imps.hpp File Reference . . . 3631
6.436.1 Detailed Description . . . 3631
6.437 cc_hash_table_map_/debug_no_store_hash_fn_imps.hpp File Reference . . . 3632
6.437.1 Detailed Description . . . 3632
6.438 gp_hash_table_map_/debug_no_store_hash_fn_imps.hpp File Reference . . . 3632
6.438.1 Detailed Description . . . 3632
6.439 cc_hash_table_map_/debug_store_hash_fn_imps.hpp File Reference . . . 3632
6.439.1 Detailed Description . . . 3632
6.440 gp_hash_table_map_/debug_store_hash_fn_imps.hpp File Reference . . . 3632

6.440.1 Detailed Description . . . 3632
6.441 entry_list_fn_imps.hpp File Reference . . . 3632
6.441.1 Detailed Description . . . 3632
6.442 cc_hash_table_map_/erase_no_store_hash_fn_imps.hpp File Reference . . . 3632
6.442.1 Detailed Description . . . 3632
6.443 gp_hash_table_map_/erase_no_store_hash_fn_imps.hpp File Reference . . . 3632
6.443.1 Detailed Description . . . 3632
6.444 cc_hash_table_map_/erase_store_hash_fn_imps.hpp File Reference . . . 3632
6.444.1 Detailed Description . . . 3632
6.445 gp_hash_table_map_/erase_store_hash_fn_imps.hpp File Reference . . . 3632
6.445.1 Detailed Description . . . 3632
6.446 cc_hash_table_map_/find_store_hash_fn_imps.hpp File Reference . . . 3632
6.446.1 Detailed Description . . . 3632
6.447 gp_hash_table_map_/find_store_hash_fn_imps.hpp File Reference . . . 3633
6.447.1 Detailed Description . . . 3633
6.448 cc_hash_table_map_/insert_no_store_hash_fn_imps.hpp File Reference . . . 3633
6.448.1 Detailed Description . . . 3633
6.449 gp_hash_table_map_/insert_no_store_hash_fn_imps.hpp File Reference . . . 3633
6.449.1 Detailed Description . . . 3633
6.450 cc_hash_table_map_/insert_store_hash_fn_imps.hpp File Reference . . . 3633
6.450.1 Detailed Description . . . 3633
6.451 gp_hash_table_map_/insert_store_hash_fn_imps.hpp File Reference . . . 3633
6.451.1 Detailed Description . . . 3633
6.452 cc_hash_table_map_/resize_fn_imps.hpp File Reference . . . 3633
6.452.1 Detailed Description . . . 3633
6.453 gp_hash_table_map_/resize_fn_imps.hpp File Reference . . . 3633
6.453.1 Detailed Description . . . 3633
6.454 cc_hash_table_map_/resize_no_store_hash_fn_imps.hpp File Reference . . . 3633
6.454.1 Detailed Description . . . 3633
6.455 gp_hash_table_map_/resize_no_store_hash_fn_imps.hpp File Reference . . . 3633
6.455.1 Detailed Description . . . 3633
6.456 cc_hash_table_map_/resize_store_hash_fn_imps.hpp File Reference . . . 3633
6.456.1 Detailed Description . . . 3633
6.457 gp_hash_table_map_/resize_store_hash_fn_imps.hpp File Reference . . . 3634
6.457.1 Detailed Description . . . 3634
6.458 size_fn_imps.hpp File Reference . . . 3634
6.458.1 Detailed Description . . . 3634
6.459 cond_dealtor.hpp File Reference . . . 3634
6.459.1 Detailed Description . . . 3634

6.460 container_base_dispatch.hpp File Reference . . . 3634
6.460.1 Detailed Description . . . 3635
6.461 debug_map_base.hpp File Reference . . . 3635
6.461.1 Detailed Description . . . 3635
6.462 eq_by_less.hpp File Reference . . . 3635
6.462.1 Detailed Description . . . 3635
6.463 hash_eq_fn.hpp File Reference . . . 3635
6.463.1 Detailed Description . . . 3635
6.464 find_no_store_hash_fn_imps.hpp File Reference . . . 3636
6.464.1 Detailed Description . . . 3636
6.465 gp_ht_map_.hpp File Reference . . . 3636
6.465.1 Detailed Description . . . 3636
6.466 iterator_fn_imps.hpp File Reference . . . 3636
6.466.1 Detailed Description . . . 3636
6.467 direct_mask_range_hashing_imp.hpp File Reference . . . 3637
6.467.1 Detailed Description . . . 3637
6.468 direct_mod_range_hashing_imp.hpp File Reference . . . 3637
6.468.1 Detailed Description . . . 3637
6.469 linear_probe_fn_imp.hpp File Reference . . . 3637
6.469.1 Detailed Description . . . 3637
6.470 mask_based_range_hashing.hpp File Reference . . . 3637
6.470.1 Detailed Description . . . 3637
6.471 mod_based_range_hashing.hpp File Reference . . . 3637
6.471.1 Detailed Description . . . 3637
6.472 probe_fn_base.hpp File Reference . . . 3638
6.472.1 Detailed Description . . . 3638
6.473 quadratic_probe_fn_imp.hpp File Reference . . . 3638
6.473.1 Detailed Description . . . 3638
6.474 ranged_hash_fn.hpp File Reference . . . 3638
6.474.1 Detailed Description . . . 3638
6.475 ranged_probe_fn.hpp File Reference . . . 3638
6.475.1 Detailed Description . . . 3639
6.476 sample_probe_fn.hpp File Reference . . . 3639
6.476.1 Detailed Description . . . 3639
6.477 sample_range_hashing.hpp File Reference . . . 3639
6.477.1 Detailed Description . . . 3639
6.478 sample_ranged_hash_fn.hpp File Reference . . . 3639
6.478.1 Detailed Description . . . 3639
6.479 sample_ranged_probe_fn.hpp File Reference . . . 3640

6.479.1 Detailed Description . . . 3640
6.480 left_child_next_sibling_heap_.hpp File Reference . . . 3640
6.480.1 Detailed Description . . . 3640
6.481 left_child_next_sibling_heap_/node.hpp File Reference . . . 3640
6.481.1 Detailed Description . . . 3640
6.482 rb_tree_map_/node.hpp File Reference . . . 3641
6.482.1 Detailed Description . . . 3641
6.483 splay_tree_/node.hpp File Reference . . . 3641
6.483.1 Detailed Description . . . 3641
6.484 entry_metadata_base.hpp File Reference . . . 3641
6.484.1 Detailed Description . . . 3641
6.485 lu_map_.hpp File Reference . . . 3641
6.485.1 Detailed Description . . . 3642
6.486 lu_counter_metadata.hpp File Reference . . . 3642
6.486.1 Detailed Description . . . 3642
6.487 sample_update_policy.hpp File Reference . . . 3642
6.487.1 Detailed Description . . . 3642
6.488 ov_tree_map_.hpp File Reference . . . 3642
6.488.1 Detailed Description . . . 3643
6.489 pairing_heap_.hpp File Reference . . . 3643
6.489.1 Detailed Description . . . 3643
6.490 insert_join_fn_imps.hpp File Reference . . . 3643
6.490.1 Detailed Description . . . 3643
6.491 pat_trie_.hpp File Reference . . . 3643
6.491.1 Detailed Description . . . 3644
6.492 pat_trie_base.hpp File Reference . . . 3644
6.492.1 Detailed Description . . . 3645
6.493 split_fn_imps.hpp File Reference . . . 3645
6.493.1 Detailed Description . . . 3645
6.494 synth_access_traits.hpp File Reference . . . 3645
6.494.1 Detailed Description . . . 3645
6.495 update_fn_imps.hpp File Reference . . . 3645
6.495.1 Detailed Description . . . 3645
6.496 priority_queue_base_dispatch.hpp File Reference . . . 3646
6.496.1 Detailed Description . . . 3646
6.497 rb_tree_.hpp File Reference . . . 3646
6.497.1 Detailed Description . . . 3646
6.498 rc.hpp File Reference . . . 3646
6.498.1 Detailed Description . . . 3647

6.499 rc_binomial_heap_.hpp File Reference . . . 3647
6.499.1 Detailed Description . . . 3647
6.500 cc_hash_max_collision_check_resize_trigger_imp.hpp File Reference . . . 3647
6.500.1 Detailed Description . . . 3647
6.501 hash_exponential_size_policy_imp.hpp File Reference . . . 3647
6.501.1 Detailed Description . . . 3647
6.502 hash_load_check_resize_trigger_imp.hpp File Reference . . . 3647
6.502.1 Detailed Description . . . 3647
6.503 hash_load_check_resize_trigger_size_base.hpp File Reference . . . 3647
6.503.1 Detailed Description . . . 3647
6.504 hash_prime_size_policy_imp.hpp File Reference . . . 3648
6.504.1 Detailed Description . . . 3648
6.505 hash_standard_resize_policy_imp.hpp File Reference . . . 3648
6.505.1 Detailed Description . . . 3648
6.506 sample_resize_policy.hpp File Reference . . . 3648
6.506.1 Detailed Description . . . 3648
6.507 sample_resize_trigger.hpp File Reference . . . 3648
6.507.1 Detailed Description . . . 3648
6.508 sample_size_policy.hpp File Reference . . . 3648
6.508.1 Detailed Description . . . 3648
6.509 splay_fn_imps.hpp File Reference . . . 3649
6.509.1 Detailed Description . . . 3649
6.510 splay_tree_.hpp File Reference . . . 3649
6.510.1 Detailed Description . . . 3649
6.511 standard_policies.hpp File Reference . . . 3649
6.511.1 Detailed Description . . . 3650
6.511.2 Enumeration Type Documentation . . . 3650
6.512 thin_heap_.hpp File Reference . . . 3650
6.512.1 Detailed Description . . . 3650
6.513 tree_policy/node_metadata_selector.hpp File Reference . . . 3650
6.513.1 Detailed Description . . . 3650
6.514 trie_policy/node_metadata_selector.hpp File Reference . . . 3651
6.514.1 Detailed Description . . . 3651
6.515 tree_policy/order_statistics_imp.hpp File Reference . . . 3651
6.515.1 Detailed Description . . . 3651
6.516 trie_policy/order_statistics_imp.hpp File Reference . . . 3651
6.516.1 Detailed Description . . . 3651
6.517 sample_tree_node_update.hpp File Reference . . . 3651
6.517.1 Detailed Description . . . 3651

6.518 tree_trace_base.hpp File Reference . . . 3651
6.518.1 Detailed Description . . . 3651
6.519 prefix_search_node_update_imp.hpp File Reference . . . 3651
6.519.1 Detailed Description . . . 3651
6.520 sample_trie_access_traits.hpp File Reference . . . 3652
6.520.1 Detailed Description . . . 3652
6.521 sample_trie_node_update.hpp File Reference . . . 3652
6.521.1 Detailed Description . . . 3652
6.522 trie_policy_base.hpp File Reference . . . 3652
6.522.1 Detailed Description . . . 3652
6.523 trie_string_access_traits_imp.hpp File Reference . . . 3652
6.523.1 Detailed Description . . . 3652
6.524 type_utils.hpp File Reference . . . 3652
6.524.1 Detailed Description . . . 3653
6.525 types_traits.hpp File Reference . . . 3653
6.525.1 Detailed Description . . . 3653
6.526 iterator.hpp File Reference . . . 3653
6.526.1 Detailed Description . . . 3653
6.527 point_iterator.hpp File Reference . . . 3654
6.527.1 Detailed Description . . . 3654
6.528 exception.hpp File Reference . . . 3654
6.528.1 Detailed Description . . . 3654
6.529 hash_policy.hpp File Reference . . . 3654
6.529.1 Detailed Description . . . 3655
6.530 list_update_policy.hpp File Reference . . . 3655
6.530.1 Detailed Description . . . 3655
6.531 priority_queue.hpp File Reference . . . 3655
6.531.1 Detailed Description . . . 3656
6.532 tag_and_trait.hpp File Reference . . . 3656
6.532.1 Detailed Description . . . 3657
6.533 tree_policy.hpp File Reference . . . 3657
6.533.1 Detailed Description . . . 3657
6.534 trie_policy.hpp File Reference . . . 3657
6.534.1 Detailed Description . . . 3658
6.535 pod_char_traits.h File Reference . . . 3658
6.535.1 Detailed Description . . . 3658
6.536 pointer.h File Reference . . . 3658
6.536.1 Detailed Description . . . 3660
6.537 pool_allocator.h File Reference . . . 3660

6.537.1 Detailed Description . . . 3660
6.538 rb_tree File Reference . . . 3660
6.538.1 Detailed Description . . . 3661
6.539 rc_string_base.h File Reference . . . 3661
6.539.1 Detailed Description . . . 3661
6.540 rope File Reference . . . 3661
6.540.1 Detailed Description . . . 3664
6.541 ropeimpl.h File Reference . . . 3664
6.541.1 Detailed Description . . . 3671
6.542 slist File Reference . . . 3671
6.542.1 Detailed Description . . . 3672
6.543 sso_string_base.h File Reference . . . 3672
6.543.1 Detailed Description . . . 3672
6.544 stdio_filebuf.h File Reference . . . 3672
6.544.1 Detailed Description . . . 3672
6.545 stdio_sync_filebuf.h File Reference . . . 3672
6.545.1 Detailed Description . . . 3673
6.546 string_conversions.h File Reference . . . 3673
6.546.1 Detailed Description . . . 3673
6.547 throw_allocator.h File Reference . . . 3673
6.547.1 Detailed Description . . . 3674
6.548 type_traits.h File Reference . . . 3674
6.548.1 Detailed Description . . . 3674
6.549 typelist.h File Reference . . . 3674
6.549.1 Detailed Description . . . 3675
6.550 vstring.h File Reference . . . 3675
6.550.1 Detailed Description . . . 3678
6.551 vstring.tcc File Reference . . . 3678
6.551.1 Detailed Description . . . 3679
6.552 vstring_fwd.h File Reference . . . 3679
6.552.1 Detailed Description . . . 3680
6.553 vstring_util.h File Reference . . . 3680
6.553.1 Detailed Description . . . 3680
6.554 fenv.h File Reference . . . 3680
6.554.1 Detailed Description . . . 3680
6.555 experimental/filesystem File Reference . . . 3680
6.555.1 Detailed Description . . . 3680
6.556 filesystem File Reference . . . 3680
6.556.1 Detailed Description . . . 3680

6.557 flat_map File Reference . . . 3681
6.557.1 Detailed Description . . . 3681
6.558 flat_set File Reference . . . 3681
6.558.1 Detailed Description . . . 3681
6.559 format File Reference . . . 3681
6.559.1 Detailed Description . . . 3681
6.560 debug/forward_list File Reference . . . 3681
6.560.1 Detailed Description . . . 3682
6.561 experimental/forward_list File Reference . . . 3682
6.561.1 Detailed Description . . . 3682
6.562 forward_list File Reference . . . 3682
6.562.1 Detailed Description . . . 3683
6.563 fstream File Reference . . . 3683
6.563.1 Detailed Description . . . 3684
6.564 experimental/functional File Reference . . . 3684
6.564.1 Detailed Description . . . 3684
6.565 ext/functional File Reference . . . 3685
6.565.1 Detailed Description . . . 3685
6.566 functional File Reference . . . 3686
6.566.1 Detailed Description . . . 3687
6.567 future File Reference . . . 3688
6.567.1 Detailed Description . . . 3689
6.568 generator File Reference . . . 3689
6.568.1 Detailed Description . . . 3689
6.569 inplace_vector File Reference . . . 3689
6.569.1 Detailed Description . . . 3689
6.570 iomanip File Reference . . . 3689
6.570.1 Detailed Description . . . 3691
6.571 ios File Reference . . . 3691
6.571.1 Detailed Description . . . 3691
6.572 iosfwd File Reference . . . 3691
6.572.1 Detailed Description . . . 3692
6.573 iostream File Reference . . . 3692
6.573.1 Detailed Description . . . 3692
6.574 istream File Reference . . . 3693
6.574.1 Detailed Description . . . 3693
6.575 experimental/iterator File Reference . . . 3694
6.575.1 Detailed Description . . . 3694
6.576 ext/iterator File Reference . . . 3694

6.576.1 Detailed Description . . . 3694
6.577 iterator File Reference . . . 3694
6.577.1 Detailed Description . . . 3695
6.578 latch File Reference . . . 3695
6.578.1 Detailed Description . . . 3695
6.579 limits File Reference . . . 3695
6.579.1 Detailed Description . . . 3696
6.580 debug/list File Reference . . . 3696
6.580.1 Detailed Description . . . 3697
6.581 experimental/list File Reference . . . 3697
6.581.1 Detailed Description . . . 3697
6.582 list File Reference . . . 3697
6.582.1 Detailed Description . . . 3698
6.583 locale File Reference . . . 3698
6.583.1 Detailed Description . . . 3698
6.584 debug/map File Reference . . . 3698
6.584.1 Detailed Description . . . 3698
6.585 experimental/map File Reference . . . 3698
6.585.1 Detailed Description . . . 3699
6.586 map File Reference . . . 3699
6.586.1 Detailed Description . . . 3699
6.587 math.h File Reference . . . 3700
6.587.1 Detailed Description . . . 3701
6.587.2 Function Documentation . . . 3701
6.588 mdspan File Reference . . . 3703
6.588.1 Detailed Description . . . 3703
6.589 experimental/memory File Reference . . . 3703
6.589.1 Detailed Description . . . 3704
6.590 ext/memory File Reference . . . 3704
6.590.1 Detailed Description . . . 3705
6.591 memory File Reference . . . 3705
6.591.1 Detailed Description . . . 3706
6.592 experimental/memory_resource File Reference . . . 3706
6.592.1 Detailed Description . . . 3706
6.592.2 Function Documentation . . . 3707
6.593 memory_resource File Reference . . . 3707
6.593.1 Detailed Description . . . 3707
6.593.2 Function Documentation . . . 3707
6.594 mutex File Reference . . . 3708

6.594.1 Detailed Description . . . 3708
6.595 numbers File Reference . . . 3708
6.595.1 Detailed Description . . . 3708
6.596 experimental/numeric File Reference . . . 3708
6.596.1 Detailed Description . . . 3709
6.597 ext/numeric File Reference . . . 3709
6.597.1 Detailed Description . . . 3709
6.598 numeric File Reference . . . 3709
6.598.1 Detailed Description . . . 3711
6.599 parallel/numeric File Reference . . . 3711
6.599.1 Detailed Description . . . 3713
6.600 experimental/optional File Reference . . . 3713
6.600.1 Detailed Description . . . 3713
6.601 optional File Reference . . . 3713
6.601.1 Detailed Description . . . 3713
6.602 ostream File Reference . . . 3713
6.602.1 Detailed Description . . . 3714
6.603 algo.h File Reference . . . 3714
6.603.1 Detailed Description . . . 3723
6.604 algobase.h File Reference . . . 3723
6.604.1 Detailed Description . . . 3725
6.605 balanced_quicksort.h File Reference . . . 3725
6.605.1 Detailed Description . . . 3726
6.606 base.h File Reference . . . 3726
6.606.1 Detailed Description . . . 3726
6.607 basic_iterator.h File Reference . . . 3727
6.607.1 Detailed Description . . . 3727
6.608 checkers.h File Reference . . . 3727
6.608.1 Detailed Description . . . 3727
6.609 compiletime_settings.h File Reference . . . 3727
6.609.1 Detailed Description . . . 3727
6.609.2 Macro Definition Documentation . . . 3727
6.610 equally_split.h File Reference . . . 3728
6.610.1 Detailed Description . . . 3729
6.611 features.h File Reference . . . 3729
6.611.1 Detailed Description . . . 3729
6.611.2 Macro Definition Documentation . . . 3729
6.612 find.h File Reference . . . 3731
6.612.1 Detailed Description . . . 3731

6.613 find_selectors.h File Reference . . . 3731
6.613.1 Detailed Description . . . 3731
6.614 for_each.h File Reference . . . 3731
6.614.1 Detailed Description . . . 3732
6.615 for_each_selectors.h File Reference . . . 3732
6.615.1 Detailed Description . . . 3732
6.616 iterator.h File Reference . . . 3732
6.616.1 Detailed Description . . . 3732
6.617 list_partition.h File Reference . . . 3733
6.617.1 Detailed Description . . . 3733
6.618 losertree.h File Reference . . . 3733
6.618.1 Detailed Description . . . 3733
6.619 merge.h File Reference . . . 3733
6.619.1 Detailed Description . . . 3734
6.620 multiseq_selection.h File Reference . . . 3734
6.620.1 Detailed Description . . . 3734
6.621 multiway_merge.h File Reference . . . 3735
6.621.1 Detailed Description . . . 3737
6.621.2 Macro Definition Documentation . . . 3737
6.622 multiway_mergesort.h File Reference . . . 3737
6.622.1 Detailed Description . . . 3738
6.623 numericfwd.h File Reference . . . 3738
6.623.1 Detailed Description . . . 3739
6.624 omp_loop.h File Reference . . . 3739
6.624.1 Detailed Description . . . 3740
6.625 omp_loop_static.h File Reference . . . 3740
6.625.1 Detailed Description . . . 3740
6.626 par_loop.h File Reference . . . 3740
6.626.1 Detailed Description . . . 3740
6.627 parallel.h File Reference . . . 3740
6.627.1 Detailed Description . . . 3740
6.628 partial_sum.h File Reference . . . 3741
6.628.1 Detailed Description . . . 3741
6.629 partition.h File Reference . . . 3741
6.629.1 Detailed Description . . . 3741
6.629.2 Macro Definition Documentation . . . 3741
6.630 queue.h File Reference . . . 3742
6.630.1 Detailed Description . . . 3742
6.630.2 Macro Definition Documentation . . . 3742

6.631 quicksort.h File Reference . . . 3742
6.631.1 Detailed Description . . . 3742
6.632 random_number.h File Reference . . . 3742
6.632.1 Detailed Description . . . 3743
6.633 random_shuffle.h File Reference . . . 3743
6.633.1 Detailed Description . . . 3743
6.634 search.h File Reference . . . 3743
6.634.1 Detailed Description . . . 3744
6.635 set_operations.h File Reference . . . 3744
6.635.1 Detailed Description . . . 3744
6.636 settings.h File Reference . . . 3744
6.636.1 Detailed Description . . . 3745
6.636.2 Deciding whether to run an algorithm in parallel. . . . 3745
6.636.3 Macro Definition Documentation . . . 3745
6.637 sort.h File Reference . . . 3745
6.637.1 Detailed Description . . . 3746
6.638 tags.h File Reference . . . 3746
6.638.1 Detailed Description . . . 3747
6.639 types.h File Reference . . . 3747
6.639.1 Detailed Description . . . 3747
6.640 unique_copy.h File Reference . . . 3747
6.640.1 Detailed Description . . . 3748
6.641 workstealing.h File Reference . . . 3748
6.641.1 Detailed Description . . . 3748
6.642 print File Reference . . . 3748
6.642.1 Detailed Description . . . 3748
6.643 queue File Reference . . . 3748
6.643.1 Detailed Description . . . 3748
6.644 experimental/random File Reference . . . 3749
6.644.1 Detailed Description . . . 3749
6.645 random File Reference . . . 3749
6.645.1 Detailed Description . . . 3749
6.646 ranges File Reference . . . 3749
6.646.1 Detailed Description . . . 3749
6.647 experimental/ratio File Reference . . . 3749
6.647.1 Detailed Description . . . 3750
6.648 ratio File Reference . . . 3750
6.648.1 Detailed Description . . . 3751
6.649 tr2/ratio File Reference . . . 3751

6.649.1 Detailed Description . . . 3751
6.650 experimental/regex File Reference . . . 3751
6.650.1 Detailed Description . . . 3752
6.651 regex File Reference . . . 3752
6.651.1 Detailed Description . . . 3752
6.652 scoped_allocator File Reference . . . 3752
6.652.1 Detailed Description . . . 3752
6.653 semaphore File Reference . . . 3753
6.653.1 Detailed Description . . . 3753
6.654 debug/set File Reference . . . 3753
6.654.1 Detailed Description . . . 3753
6.655 experimental/set File Reference . . . 3753
6.655.1 Detailed Description . . . 3753
6.656 set File Reference . . . 3754
6.656.1 Detailed Description . . . 3754
6.657 shared_mutex File Reference . . . 3754
6.657.1 Detailed Description . . . 3754
6.658 source_location File Reference . . . 3755
6.658.1 Detailed Description . . . 3755
6.659 span File Reference . . . 3755
6.659.1 Detailed Description . . . 3755
6.660 sstream File Reference . . . 3755
6.660.1 Detailed Description . . . 3756
6.661 stack File Reference . . . 3756
6.661.1 Detailed Description . . . 3756
6.662 stdatomic.h File Reference . . . 3756
6.662.1 Detailed Description . . . 3756
6.663 stdbit.h File Reference . . . 3756
6.663.1 Detailed Description . . . 3756
6.664 stdckdint.h File Reference . . . 3756
6.664.1 Detailed Description . . . 3756
6.665 stdexcept File Reference . . . 3756
6.665.1 Detailed Description . . . 3757
6.666 stdlib.h File Reference . . . 3757
6.666.1 Detailed Description . . . 3757
6.666.2 Function Documentation . . . 3757
6.667 stop_token File Reference . . . 3757
6.667.1 Detailed Description . . . 3757
6.668 streambuf File Reference . . . 3757

6.668.1 Detailed Description . . . 3758
6.669 debug/string File Reference . . . 3758
6.669.1 Detailed Description . . . 3760
6.670 experimental/string File Reference . . . 3760
6.670.1 Detailed Description . . . 3761
6.671 string File Reference . . . 3761
6.671.1 Detailed Description . . . 3761
6.672 experimental/string_view File Reference . . . 3761
6.672.1 Detailed Description . . . 3763
6.673 string_view File Reference . . . 3763
6.673.1 Detailed Description . . . 3764
6.674 syncstream File Reference . . . 3764
6.674.1 Detailed Description . . . 3764
6.675 experimental/system_error File Reference . . . 3764
6.675.1 Detailed Description . . . 3765
6.676 system_error File Reference . . . 3765
6.676.1 Detailed Description . . . 3765
6.677 text_encoding File Reference . . . 3765
6.677.1 Detailed Description . . . 3765
6.678 tgmath.h File Reference . . . 3765
6.678.1 Detailed Description . . . 3765
6.679 thread File Reference . . . 3766
6.679.1 Detailed Description . . . 3766
6.680 bool_set File Reference . . . 3766
6.680.1 Detailed Description . . . 3767
6.681 bool_set.tcc File Reference . . . 3767
6.681.1 Detailed Description . . . 3767
6.682 dynamic_bitset File Reference . . . 3767
6.682.1 Detailed Description . . . 3768
6.683 dynamic_bitset.tcc File Reference . . . 3768
6.683.1 Detailed Description . . . 3768
6.684 experimental/tuple File Reference . . . 3768
6.684.1 Detailed Description . . . 3769
6.685 tuple File Reference . . . 3769
6.685.1 Detailed Description . . . 3771
6.686 experimental/type_traits File Reference . . . 3771
6.686.1 Detailed Description . . . 3774
6.687 tr2/type_traits File Reference . . . 3774
6.687.1 Detailed Description . . . 3775

6.688 type_traits File Reference . . . 3775
6.688.1 Detailed Description . . . 3779
6.689 typeindex File Reference . . . 3779
6.689.1 Detailed Description . . . 3779
6.690 debug/unordered_map File Reference . . . 3779
6.690.1 Detailed Description . . . 3781
6.691 experimental/unordered_map File Reference . . . 3781
6.691.1 Detailed Description . . . 3782
6.692 unordered_map File Reference . . . 3782
6.692.1 Detailed Description . . . 3782
6.693 debug/unordered_set File Reference . . . 3782
6.693.1 Detailed Description . . . 3784
6.694 experimental/unordered_set File Reference . . . 3784
6.694.1 Detailed Description . . . 3785
6.695 unordered_set File Reference . . . 3785
6.695.1 Detailed Description . . . 3785
6.696 experimental/utility File Reference . . . 3786
6.696.1 Detailed Description . . . 3786
6.697 utility File Reference . . . 3786
6.697.1 Detailed Description . . . 3786
6.698 valarray File Reference . . . 3786
6.698.1 Detailed Description . . . 3787
6.699 variant File Reference . . . 3787
6.699.1 Detailed Description . . . 3787
6.700 debug/vector File Reference . . . 3787
6.700.1 Detailed Description . . . 3788
6.701 experimental/vector File Reference . . . 3788
6.701.1 Detailed Description . . . 3788
6.702 vector File Reference . . . 3789
6.702.1 Detailed Description . . . 3789
6.703 atomic_word.h File Reference . . . 3789
6.703.1 Detailed Description . . . 3789
6.704 basic_file.h File Reference . . . 3789
6.704.1 Detailed Description . . . 3789
6.705 c++allocator.h File Reference . . . 3789
6.705.1 Detailed Description . . . 3790
6.706 c++config.h File Reference . . . 3790
6.706.1 Detailed Description . . . 3796
6.707 c++io.h File Reference . . . 3797

6.707.1 Detailed Description . . . 3797
6.708 c++locale.h File Reference . . . 3797
6.708.1 Detailed Description . . . 3797
6.709 c++locale_internal.h File Reference . . . 3797
6.709.1 Detailed Description . . . 3797
6.710 parallel/compatibility.h File Reference . . . 3798
6.710.1 Detailed Description . . . 3798
6.711 x86_64-pc-linux-gnu/bits/compatibility.h File Reference . . . 3798
6.711.1 Detailed Description . . . 3798
6.712 cpu_defines.h File Reference . . . 3798
6.712.1 Detailed Description . . . 3798
6.713 ctype_base.h File Reference . . . 3798
6.713.1 Detailed Description . . . 3798
6.714 ctype_inline.h File Reference . . . 3798
6.714.1 Detailed Description . . . 3799
6.715 cxxabi_tweaks.h File Reference . . . 3799
6.715.1 Detailed Description . . . 3799
6.716 error_constants.h File Reference . . . 3799
6.716.1 Detailed Description . . . 3800
6.717 extc++.h File Reference . . . 3800
6.717.1 Detailed Description . . . 3800
6.718 messages_members.h File Reference . . . 3800
6.718.1 Detailed Description . . . 3800
6.719 opt_random.h File Reference . . . 3800
6.719.1 Detailed Description . . . 3800
6.720 os_defines.h File Reference . . . 3800
6.720.1 Detailed Description . . . 3800
6.721 stdc++.h File Reference . . . 3800
6.721.1 Detailed Description . . . 3800
6.722 stdtr1c++.h File Reference . . . 3800
6.722.1 Detailed Description . . . 3800
6.723 time_members.h File Reference . . . 3801
6.723.1 Detailed Description . . . 3801
6.724 compare File Reference . . . 3801
6.724.1 Detailed Description . . . 3802
6.725 cxxabi.h File Reference . . . 3802
6.725.1 Detailed Description . . . 3803
6.725.2 Function Documentation . . . 3803
6.726 exception File Reference . . . 3804

6.726.1 Detailed Description . . . 3804
6.727 initializer_list File Reference . . . 3805
6.727.1 Detailed Description . . . 3805
6.728 new File Reference . . . 3805
6.728.1 Detailed Description . . . 3806
6.728.2 Function Documentation . . . 3806
6.729 typeinfo File Reference . . . 3806
6.729.1 Detailed Description . . . 3807

Index 3809

1 Deprecated List

Module negators

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

Module pointer_adaptors

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

Module ptrmem_adaptors

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

Struct std::aligned_storage< _Len, _Align >

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

Struct std::aligned_union< _Len, _Types >

Deprecated in C++23.

Class std::auto_ptr< _Tp >

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

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

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

Member std::get_unexpected () noexcept

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

Struct std::is_literal_type< _Tp >

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

Struct std::is_pod< _Tp >

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

Struct std::is_trivial< _Tp >

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

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

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

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

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

Member std::set_unexpected (unexpected_handler) noexcept

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

Struct std::unary_function< _Arg, _Result >

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

Member std::unexpected ()

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

2 Todo List

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Member __gnu_cxx::random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out_first, _RandomAccessIterator __out_last)

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

Member __gnu_cxx::random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out_first, _RandomAccessIterator __out_last, _RandomNumberGenerator &__rand)

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

Member __gnu_cxx::random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const _Distance __n)

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

Member __gnu_cxx::random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const _Distance __n, _RandomNumberGenerator &__rand)

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

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

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

Class __gnu_cxx::rope< _CharT, _Alloc >

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

Class __gnu_cxx::slist< _Tp, _Alloc >

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

Module mathsf

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

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

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

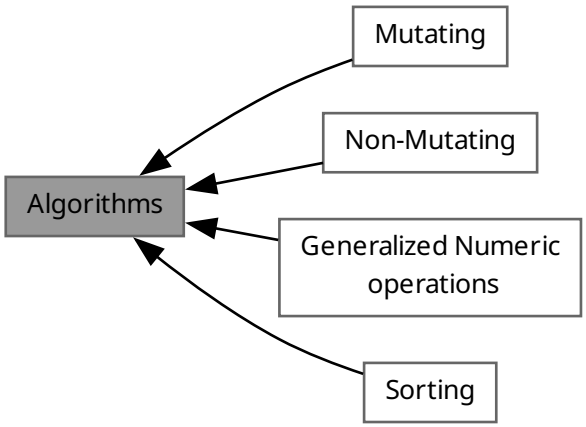
Member std::regex_traits< _Ch_type >::transform_primary (_Fwd_iter __first, _Fwd_iter __last) const

Implement this function correctly.

3 Topic Documentation

3.1 Algorithms

Collaboration diagram for Algorithms:



Topics

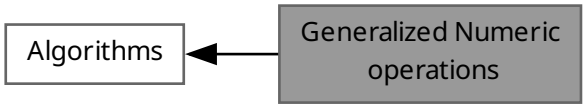
- Generalized Numeric operations
- Mutating
- Non-Mutating
- Sorting

3.1.1 Detailed Description

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

3.1.2 Generalized Numeric operations

Collaboration diagram for Generalized Numeric operations:



Functions

- template<typename _InputIterator, typename _Tp>
  constexpr _Tp std::accumulate (_InputIterator __first, _InputIterator __last, _Tp __init)
- template<typename _InputIterator, typename _Tp, typename _BinaryOperation>
  constexpr _Tp std::accumulate (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __↩
  binary_op)
- template<typename _InputIterator, typename _OutputIterator>
  constexpr _OutputIterator std::adjacent_difference (_InputIterator __first, _InputIterator __last, _OutputIterator ↩
  __result)
- template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation>
  constexpr _OutputIterator std::adjacent_difference (_InputIterator __first, _InputIterator __last, _OutputIterator ↩
  __result, _BinaryOperation __binary_op)
- template<typename _InputIterator, typename _OutputIterator, typename _Tp>
  constexpr _OutputIterator std::exclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __↩
  result, _Tp __init)
- template<typename _InputIterator, typename _OutputIterator, typename _Tp, typename _BinaryOperation>
  constexpr _OutputIterator std::exclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __↩
  result, _Tp __init, _BinaryOperation __binary_op)
- template<typename _InputIterator, typename _OutputIterator>
  constexpr _OutputIterator std::inclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __↩
  result)
- template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation>
  constexpr _OutputIterator std::inclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __↩
  result, _BinaryOperation __binary_op)
- template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation, typename _Tp>
  constexpr _OutputIterator std::inclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __↩
  result, _BinaryOperation __binary_op, _Tp __init)
- template<typename _InputIterator1, typename _InputIterator2, typename _Tp>
  constexpr _Tp std::inner_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp
  __init)
- template<typename _InputIterator1, typename _InputIterator2, typename _Tp, typename _BinaryOperation1, typename _Binary↩
  Operation2>
  constexpr _Tp std::inner_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp
  __init, _BinaryOperation1 __binary_op1, _BinaryOperation2 __binary_op2)

- template<typename _ForwardIterator, typename _Tp>
  constexpr void std::iota (_ForwardIterator __first, _ForwardIterator __last, _Tp __value)
- template<typename _InputIterator, typename _OutputIterator>
  constexpr _OutputIterator std::partial_sum (_InputIterator __first, _InputIterator __last, _OutputIterator __result)
- template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation>
  constexpr _OutputIterator std::partial_sum (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op)
- template<typename _InputIterator>
  constexpr iterator_traits< _InputIterator >::value_type std::reduce (_InputIterator __first, _InputIterator __last)
- template<typename _InputIterator, typename _Tp>
  constexpr _Tp std::reduce (_InputIterator __first, _InputIterator __last, _Tp __init)
- template<typename _InputIterator, typename _Tp, typename _BinaryOperation>
  constexpr _Tp std::reduce (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __binary_op)
- template<typename _InputIterator, typename _OutputIterator, typename _Tp, typename _BinaryOperation, typename _UnaryOperation>
  constexpr _OutputIterator std::transform_exclusive_scan (_InputIterator __first, _InputIterator __last, _Output↩ Iterator __result, _Tp __init, _BinaryOperation __binary_op, _UnaryOperation __unary_op)
- template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation, typename _UnaryOperation>
  constexpr _OutputIterator std::transform_inclusive_scan (_InputIterator __first, _InputIterator __last, _Output↩ Iterator __result, _BinaryOperation __binary_op, _UnaryOperation __unary_op)
- template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation, typename _UnaryOperation, typename _Tp>
  constexpr _OutputIterator std::transform_inclusive_scan (_InputIterator __first, _InputIterator __last, _Output↩ Iterator __result, _BinaryOperation __binary_op, _UnaryOperation __unary_op, _Tp __init)
- template<typename _InputIterator, typename _Tp, typename _BinaryOperation, typename _UnaryOperation>
  constexpr _Tp std::transform_reduce (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __binary_op, _UnaryOperation __unary_op)
- template<typename _InputIterator1, typename _InputIterator2, typename _Tp>
  constexpr _Tp std::transform_reduce (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init)
- template<typename _InputIterator1, typename _InputIterator2, typename _Tp, typename _BinaryOperation1, typename _Binary↩ Operation2>
  constexpr _Tp std::transform_reduce (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init, _BinaryOperation1 __binary_op1, _BinaryOperation2 __binary_op2)

3.1.2.1 Detailed Description

3.1.2.2 Function Documentation

accumulate() [1/2]

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

Accumulate values in a range.

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

Parameters

__first	Start of range.
__last	End of range.
__init	Starting value to add other values to.

Returns

The final sum.

accumulate() [2/2]

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

Accumulate values in a range with operation.

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

Parameters

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

Returns

The final sum.

adjacent_difference() [1/2]

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

Return differences between adjacent values.

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

Parameters

__first	Start of input range.
__last	End of input range.
__result	Output sums.

Returns

Iterator pointing just beyond the values written to result.

adjacent_difference() [2/2]

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

Return differences between adjacent values.

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

Parameters

__first	Start of input range.
__last	End of input range.
__result	Output sum.
__binary_op	Function object.

Returns

Iterator pointing just beyond the values written to result.

exclusive_scan() [1/2]

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

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

Parameters

__first	Start of input range.
__last	End of input range.
__result	Start of output range.
__init	Initial value.

Returns

The end of the output range.

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

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

exclusive_scan() [2/2]

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

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

Parameters

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

Returns

The end of the output range.

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

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

inclusive_scan() [1/3]

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

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

Parameters

__first	Start of input range.
__last	End of input range.
__result	Start of output range.

Returns

The end of the output range.

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

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

inclusive_scan() [2/3]

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

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

Parameters

__first	Start of input range.
__last	End of input range.
__result	Start of output range.
__binary_op	Function to perform summation.

Returns

The end of the output range.

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

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

inclusive_scan() [3/3]

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

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

Parameters

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

Returns

The end of the output range.

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

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

inner_product() [1/2]

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

Compute inner product of two ranges.

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

Parameters

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

Returns

The final inner product.

inner_product() [2/2]

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

Compute inner product of two ranges.

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

Parameters

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

Returns

The final inner product.

iota()

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

Create a range of sequentially increasing values.

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

Parameters

__first	Start of range.
__last	End of range.
__value	Starting value.

Returns

Nothing.

partial_sum() [1/2]

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

Return list of partial sums.

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

Parameters

__first	Start of input range.
__last	End of input range.
__result	Output sum.

Returns

Iterator pointing just beyond the values written to __result.

Referenced by __gnu_parallel::__parallel_random_shuffle_drs_pu(), and __gnu_parallel::__sequential_random_shuffle().

partial_sum() [2/2]

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

Return list of partial sums.

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

Parameters

__first	Start of input range.
__last	End of input range.
__result	Output sum.
__binary_op	Function object.

Returns

Iterator pointing just beyond the values written to __result.

reduce() [1/3]

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

Calculate reduction of values in a range.

Parameters

__first	Start of range.
__last	End of range.

Returns

The final sum.

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

reduce() [2/3]

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

Calculate reduction of values in a range.

Parameters

__first	Start of range.
__last	End of range.
__init	Starting value to add other values to.

Returns

The final sum.

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

reduce() [3/3]

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

Calculate reduction of values in a range.

Parameters

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

Returns

The final sum.

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

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

transform_exclusive_scan()

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

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

Parameters

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

Returns

The end of the output range.

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

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

transform_inclusive_scan() [1/2]

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

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

Parameters

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

Returns

The end of the output range.

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

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

transform_inclusive_scan() [2/2]

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

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

Parameters

__first	Start of input range.

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

Returns

The end of the output range.

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

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

transform_reduce() [1/3]

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

Transform the elements of a range and reduce.

Parameters

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

Returns

The final sum.

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

transform_reduce() [2/3]

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

Combine elements from two ranges and reduce.

Parameters

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

Returns

The final sum.

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

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

transform_reduce() [3/3]

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

Combine elements from two ranges and reduce.

Parameters

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

Returns

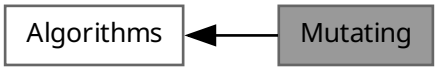
The final sum.

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

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

3.1.3 Mutating

Collaboration diagram for Mutating:



Functions

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

- template<typename _ForwardIterator, typename _Predicate>
  constexpr _ForwardIterator std::partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)
- template<typename _InputIterator, typename _OutputIterator1, typename _OutputIterator2, typename _Predicate>
  constexpr pair< _OutputIterator1, _OutputIterator2 > std::partition_copy (_InputIterator __first, _InputIterator __last, _OutputIterator1 __out_true, _OutputIterator2 __out_false, _Predicate __pred)
- template<typename _ForwardIterator, typename _Predicate>
  constexpr _ForwardIterator std::partition_point (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)
- template<typename _RandomAccessIterator>
  void std::random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _RandomNumberGenerator>
  void std::random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomNumberGenerator &&__rand)
- template<typename _ForwardIterator, typename _Tp>
  constexpr _ForwardIterator std::remove (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)
- template<typename _InputIterator, typename _OutputIterator, typename _Tp>
  constexpr _OutputIterator std::remove_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp &__value)
- template<typename _InputIterator, typename _OutputIterator, typename _Predicate>
  constexpr _OutputIterator std::remove_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)
- template<typename _ForwardIterator, typename _Predicate>
  constexpr _ForwardIterator std::remove_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)
- template<typename _ForwardIterator, typename _Tp>
  constexpr void std::replace (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__old_value, const _Tp &__new_value)
- template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _Tp>
  constexpr _OutputIterator std::replace_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred, const _Tp &__new_value)
- template<typename _ForwardIterator, typename _Predicate, typename _Tp>
  constexpr void std::replace_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, const _Tp &__new_value)
- template<typename _BidirectionalIterator>
  constexpr void std::reverse (_BidirectionalIterator __first, _BidirectionalIterator __last)
- template<typename _BidirectionalIterator, typename _OutputIterator>
  constexpr _OutputIterator std::reverse_copy (_BidirectionalIterator __first, _BidirectionalIterator __last, _OutputIterator __result)
- template<typename _ForwardIterator>
  constexpr _ForwardIterator std::rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _OutputIterator>
  constexpr _OutputIterator std::rotate_copy (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last, _OutputIterator __result)
- template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator>
  void std::shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _UniformRandomNumberGenerator &&__g)
- template<typename _ForwardIterator, typename _Predicate>
  _ForwardIterator std::stable_partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)
- template<typename _ForwardIterator1, typename _ForwardIterator2>
  constexpr _ForwardIterator2 std::swap_ranges (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)
- template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation>
  constexpr _OutputIterator std::transform (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _UnaryOperation __unary_op)

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

3.1.3.1 Detailed Description

3.1.3.2 Function Documentation

copy()

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

Copies the range [first,last) into result.

Parameters

__first	An input iterator.
__last	An input iterator.
__result	An output iterator.

Returns

result + (last - first)

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

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

copy_backward()

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

Copies the range [first,last) into result.

Parameters

__first	A bidirectional iterator.
__last	A bidirectional iterator.
__result	A bidirectional iterator.

Returns

result - (last - first)

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

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

copy_if()

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

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

Parameters

__first	An input iterator.
__last	An input iterator.
__result	An output iterator.
__pred	A predicate.

Returns

An iterator designating the end of the resulting sequence.

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

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

copy_n()

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

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

Parameters

__first	An input iterator.
__n	The number of elements to copy.
__result	An output iterator.

Returns

result+n.

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

References move().

fill()

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

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

Parameters

__first	A forward iterator.
__last	A forward iterator.
__value	A reference-to-const of arbitrary type.

Returns

Nothing.

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

fill_n()

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

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

Parameters

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

Returns

The iterator at first+n.

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

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

References __iterator_category().

generate()

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

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

Parameters

__first	A forward iterator.
__last	A forward iterator.
__gen	A function object callable with no arguments.

Returns

generate() returns no value.

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

generate_n()

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

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

Parameters

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

Returns

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

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

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

is_partitioned()

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

Checks whether the sequence is partitioned.

Parameters

__first	An input iterator.
__last	An input iterator.
__pred	A predicate.

Returns

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

iter_swap()

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

Swaps the contents of two iterators.

Parameters

_↩ _*a*	An iterator.
_↩ _*b*	Another iterator.

Returns

Nothing.

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

move()

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

Moves the range [first,last) into result.

Parameters

__first	An input iterator.
__last	An input iterator.
__result	An output iterator.

Returns

result + (last - first)

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

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

move_backward()

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

Moves the range [first,last) into result.

Parameters

__first	A bidirectional iterator.
__last	A bidirectional iterator.
__result	A bidirectional iterator.

Returns

result - (last - first)

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

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

partition()

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

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

Parameters

__first	A forward iterator.
__last	A forward iterator.
__pred	A predicate functor.

Returns

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

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

References __iterator_category(), and __partition().

partition_copy()

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

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

Parameters

__first	An input iterator.
__last	An input iterator.
__out_true	An output iterator.
__out_false	An output iterator.
__pred	A predicate.

Returns

A pair designating the ends of the resulting sequences.

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

partition_point()

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

Find the partition point of a partitioned range.

Parameters

__first	An iterator.
__last	Another iterator.
__pred	A predicate.

Returns

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

References advance(), and distance().

random_shuffle() [1/2]

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

Randomly shuffle the elements of a sequence.

Parameters

__first	A forward iterator.
__last	A forward iterator.

Returns

Nothing.

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

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

random_shuffle() [2/2]

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

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

Parameters

__first	A forward iterator.
__last	A forward iterator.
__rand	The RNG functor or function.

Returns

Nothing.

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

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

remove()

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

Remove elements from a sequence.

Parameters

__first	An input iterator.
__last	An input iterator.
__value	The value to be removed.

Returns

An iterator designating the end of the resulting sequence.

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

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

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

remove_copy()

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

Copy a sequence, removing elements of a given value.

Parameters

__first	An input iterator.
__last	An input iterator.
__result	An output iterator.
__value	The value to be removed.

Returns

An iterator designating the end of the resulting sequence.

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

remove_copy_if()

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

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

Parameters

__first	An input iterator.
__last	An input iterator.
__result	An output iterator.
__pred	A predicate.

Returns

An iterator designating the end of the resulting sequence.

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

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

remove_if()

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

Remove elements from a sequence using a predicate.

Parameters

__first	A forward iterator.
__last	A forward iterator.
__pred	A predicate.

Returns

An iterator designating the end of the resulting sequence.

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

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

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

replace()

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

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

Parameters

__first	A forward iterator.
__last	A forward iterator.
__old_value	The value to be replaced.
__new_value	The replacement value.

Returns

replace() returns no value.

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

replace_copy_if()

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

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

Parameters

__first	An input iterator.
__last	An input iterator.
__result	An output iterator.
__pred	A predicate.
__new_value	The replacement value.

Returns

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

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

replace_if()

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

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

Parameters

__first	A forward iterator.
__last	A forward iterator.
__pred	A predicate.
__new_value	The replacement value.

Returns

replace_if() returns no value.

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

reverse()

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

Reverse a sequence.

Parameters

__first	A bidirectional iterator.
__last	A bidirectional iterator.

Returns

reverse() returns no value.

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

References __iterator_category(), and __reverse().

reverse_copy()

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

Copy a sequence, reversing its elements.

Parameters

__first	A bidirectional iterator.
__last	A bidirectional iterator.
__result	An output iterator.

Returns

An iterator designating the end of the resulting sequence.

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

rotate()

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

Rotate the elements of a sequence.

Parameters

__first	A forward iterator.
__middle	A forward iterator.
__last	A forward iterator.

Returns

first + (last - middle).

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

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

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

References __iterator_category(), and __rotate().

rotate_copy()

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

Copy a sequence, rotating its elements.

Parameters

__first	A forward iterator.
__middle	A forward iterator.
__last	A forward iterator.
__result	An output iterator.

Returns

An iterator designating the end of the resulting sequence.

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

Returns

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

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

shuffle()

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

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

Parameters

__first	A forward iterator.
__last	A forward iterator.
__g	A UniformRandomNumberGenerator (26.5.1.3).

Returns

Nothing.

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

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

stable_partition()

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

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

Parameters

__first	A forward iterator.
__last	A forward iterator.
__pred	A predicate functor.

Returns

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

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

swap_ranges()

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

Swap the elements of two sequences.

Parameters

__first1	A forward iterator.
__last1	A forward iterator.
__first2	A forward iterator.

Returns

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

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

transform() [1/2]

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

Perform an operation on a sequence.

Parameters

__first	An input iterator.
__last	An input iterator.
__result	An output iterator.
__unary_op	A unary operator.

Returns

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

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

unary_op must not alter its argument.

transform() [2/2]

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

Perform an operation on corresponding elements of two sequences.

Parameters

__first1	An input iterator.
__last1	An input iterator.
__first2	An input iterator.
__result	An output iterator.
__binary_op	A binary operator.

Returns

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

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

binary_op must not alter either of its arguments.

unique() [1/2]

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

Remove consecutive duplicate values from a sequence.

Parameters

__first	A forward iterator.
__last	A forward iterator.

Returns

An iterator designating the end of the resulting sequence.

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

unique() [2/2]

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

Remove consecutive values from a sequence using a predicate.

Parameters

__first	A forward iterator.
__last	A forward iterator.
__binary_pred	A binary predicate.

Returns

An iterator designating the end of the resulting sequence.

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

unique_copy() [1/2]

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

Copy a sequence, removing consecutive duplicate values.

Parameters

__first	An input iterator.
__last	An input iterator.
__result	An output iterator.

Returns

An iterator designating the end of the resulting sequence.

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

References __iterator_category().

unique_copy() [2/2]

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

Copy a sequence, removing consecutive values using a predicate.

Parameters

__first	An input iterator.
__last	An input iterator.
__result	An output iterator.
__binary_pred	A binary predicate.

Returns

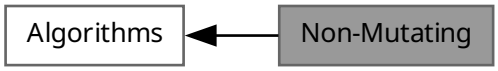
An iterator designating the end of the resulting sequence.

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

References __iterator_category().

3.1.4 Non-Mutating

Collaboration diagram for Non-Mutating:



Functions

- template<typename _ForwardIterator>
  constexpr _ForwardIterator std::adjacent_find (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _BinaryPredicate>
  constexpr _ForwardIterator std::adjacent_find (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)
- template<typename _InputIterator, typename _Predicate>
  constexpr bool std::all_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _InputIterator, typename _Predicate>
  constexpr bool std::any_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _InputIterator, typename _Tp>
  constexpr iterator_traits< _InputIterator >::difference_type std::count (_InputIterator __first, _InputIterator __last, const _Tp &__value)
- template<typename _InputIterator, typename _Predicate>
  constexpr iterator_traits< _InputIterator >::difference_type std::count_if (_InputIterator __first, _InputIterator _↩ _last, _Predicate __pred)
- template<typename _II1, typename _II2>
  constexpr bool std::equal (_II1 __first1, _II1 __last1, _II2 __first2)
- template<typename _II1, typename _II2>
  constexpr bool std::equal (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)
- template<typename _IIter1, typename _IIter2, typename _BinaryPredicate>
  constexpr bool std::equal (_IIter1 __first1, _IIter1 __last1, _IIter2 __first2, _BinaryPredicate __binary_pred)
- template<typename _IIter1, typename _IIter2, typename _BinaryPredicate>
  constexpr bool std::equal (_IIter1 __first1, _IIter1 __last1, _IIter2 __first2, _IIter2 __last2, _BinaryPredicate __↩ binary_pred)
- template<typename _InputIterator, typename _Tp>
  constexpr _InputIterator std::find (_InputIterator __first, _InputIterator __last, const _Tp &__val)
- template<typename _ForwardIterator1, typename _ForwardIterator2>
  constexpr _ForwardIterator1 std::find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _Forward↩ Iterator2 __first2, _ForwardIterator2 __last2)
- template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>
  constexpr _ForwardIterator1 std::find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _Forward↩ Iterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __comp)
- template<typename _InputIterator, typename _ForwardIterator>
  constexpr _InputIterator std::find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2)

- template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate>
  constexpr _InputIterator std::find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2, _BinaryPredicate __comp)
- template<typename _InputIterator, typename _Predicate>
  constexpr _InputIterator std::find_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _InputIterator, typename _Predicate>
  constexpr _InputIterator std::find_if_not (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _InputIterator, typename _Function>
  constexpr _Function std::for_each (_InputIterator __first, _InputIterator __last, _Function __f)
- template<typename _InputIterator, typename _Size, typename _Function>
  constexpr _InputIterator std::for_each_n (_InputIterator __first, _Size __n, _Function __f)
- template<typename _ForwardIterator1, typename _ForwardIterator2>
  constexpr bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 _↩ _first2)
- template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>
  constexpr bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 _↩ _first2, _BinaryPredicate __pred)
- template<typename _ForwardIterator1, typename _ForwardIterator2>
  constexpr bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 _↩ _first2, _ForwardIterator2 __last2)
- template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>
  constexpr bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 _↩ _first2, _ForwardIterator2 __last2, _BinaryPredicate __pred)
- template<typename _InputIterator1, typename _InputIterator2>
  constexpr pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2)
- template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>
  constexpr pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_pred)
- template<typename _InputIterator1, typename _InputIterator2>
  constexpr pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)
- template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>
  constexpr pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _BinaryPredicate __binary_pred)
- template<typename _InputIterator, typename _Predicate>
  constexpr bool std::none_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _ForwardIterator1, typename _ForwardIterator2>
  constexpr _ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)
- template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>
  constexpr _ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __predicate)
- template<typename _ForwardIterator, typename _Integer, typename _Tp>
  constexpr _ForwardIterator std::search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val)
- template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate>
  constexpr _ForwardIterator std::search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)

3.1.4.1 Detailed Description

3.1.4.2 Function Documentation

adjacent_find() [1/2]

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

Find two adjacent values in a sequence that are equal.

Parameters

__first	A forward iterator.
__last	A forward iterator.

Returns

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

adjacent_find() [2/2]

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

Find two adjacent values in a sequence using a predicate.

Parameters

__first	A forward iterator.
__last	A forward iterator.
__binary_pred	A binary predicate.

Returns

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

all_of()

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

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

Parameters

__first	An input iterator.
__last	An input iterator.
__pred	A predicate.

Returns

True if the check is true, false otherwise.

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

any_of()

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

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

Parameters

__first	An input iterator.
__last	An input iterator.
__pred	A predicate.

Returns

True if the check is true, false otherwise.

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

count()

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

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

Parameters

__first	An input iterator.
__last	An input iterator.
__value	The value to be counted.

Returns

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

count_if()

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

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

Parameters

__first	An input iterator.
__last	An input iterator.
__pred	A predicate.

Returns

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

equal() [1/4]

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

Tests a range for element-wise equality.

Parameters

__first1	An input iterator.
__last1	An input iterator.
__first2	An input iterator.

Returns

A boolean true or false.

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

equal() [2/4]

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

Tests a range for element-wise equality.

Parameters

__first1	An input iterator.
__last1	An input iterator.
__first2	An input iterator.
__last2	An input iterator.

Returns

A boolean true or false.

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

equal() [3/4]

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

Tests a range for element-wise equality.

Parameters

__first1	An input iterator.
__last1	An input iterator.
__first2	An input iterator.
__binary_pred	A binary predicate functor.

Returns

A boolean true or false.

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

equal() [4/4]

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

Tests a range for element-wise equality.

Parameters

__first1	An input iterator.
__last1	An input iterator.
__first2	An input iterator.
__last2	An input iterator.
__binary_pred	A binary predicate functor.

Returns

A boolean true or false.

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

find()

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

Find the first occurrence of a value in a sequence.

Parameters

__first	An input iterator.
__last	An input iterator.
__val	The value to find.

Returns

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

References to_address().

find_end() [1/2]

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

Find last matching subsequence in a sequence.

Parameters

__first1	Start of range to search.
__last1	End of range to search.
__first2	Start of sequence to match.
__last2	End of sequence to match.

Returns

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

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

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

References __iterator_category().

find_end() [2/2]

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

Find last matching subsequence in a sequence using a predicate.

Parameters

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

Returns

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

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

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

References __iterator_category().

find_first_of() [1/2]

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

Find element from a set in a sequence.

Parameters

__first1	Start of range to search.
__last1	End of range to search.
__first2	Start of match candidates.
__last2	End of match candidates.

Returns

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

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

find_first_of() [2/2]

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

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

Parameters

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

Returns

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

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

find_if()

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

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

Parameters

__first	An input iterator.
__last	An input iterator.
__pred	A predicate.

Returns

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

find_if_not()

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

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

Parameters

__first	An input iterator.
__last	An input iterator.
__pred	A predicate.

Returns

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

References __find_if_not().

for_each()

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

Apply a function to every element of a sequence.

Parameters

__first	An input iterator.
__last	An input iterator.
__f	A unary function object.

Returns

`__f`

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

for_each_n()

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

Apply a function to every element of a sequence.

Parameters

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

Returns

`__first+__n`

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

References move().

is_permutation() [1/4]

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

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

Parameters

__first1	Start of first range.
__last1	End of first range.
__first2	Start of second range.

Returns

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

is_permutation() [2/4]

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

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

Parameters

__first1	Start of first range.
__last1	End of first range.
__first2	Start of second range.
__pred	A binary predicate.

Returns

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

is_permutation() [3/4]

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

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

Parameters

__first1	Start of first range.
__last1	End of first range.
__first2	Start of second range.
__last2	End of first range.

Returns

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

is_permutation() [4/4]

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

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

Parameters

__first1	Start of first range.
__last1	End of first range.
__first2	Start of second range.
__last2	End of first range.
__pred	A binary predicate.

Returns

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

mismatch() [1/4]

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

Finds the places in ranges which don't match.

Parameters

__first1	An input iterator.
__last1	An input iterator.
__first2	An input iterator.

Returns

A pair of iterators pointing to the first mismatch.

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

mismatch() [2/4]

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

Finds the places in ranges which don't match.

Parameters

__first1	An input iterator.
__last1	An input iterator.
__first2	An input iterator.
__binary_pred	A binary predicate functor.

Returns

A pair of iterators pointing to the first mismatch.

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

mismatch() [3/4]

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

Finds the places in ranges which don't match.

Parameters

__first1	An input iterator.
__last1	An input iterator.
__first2	An input iterator.
__last2	An input iterator.

Returns

A pair of iterators pointing to the first mismatch.

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

mismatch() [4/4]

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

Finds the places in ranges which don't match.

Parameters

__first1	An input iterator.
__last1	An input iterator.
__first2	An input iterator.
__last2	An input iterator.
__binary_pred	A binary predicate functor.

Returns

A pair of iterators pointing to the first mismatch.

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

none_of()

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

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

Parameters

__first	An input iterator.
__last	An input iterator.
__pred	A predicate.

Returns

True if the check is true, false otherwise.

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

search() [1/2]

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

Search a sequence for a matching sub-sequence.

Parameters

__first1	A forward iterator.
__last1	A forward iterator.
__first2	A forward iterator.
__last2	A forward iterator.

Returns

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

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

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

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

search() [2/2]

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

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

Parameters

__first1	A forward iterator.
__last1	A forward iterator.
__first2	A forward iterator.
__last2	A forward iterator.
__predicate	A binary predicate.

Returns

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

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

See also

search(_ForwardIter1, _ForwardIter1, _ForwardIter2, _ForwardIter2)

search_n() [1/2]

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

Search a sequence for a number of consecutive values.

Parameters

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

Returns

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

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

search_n() [2/2]

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

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

Parameters

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

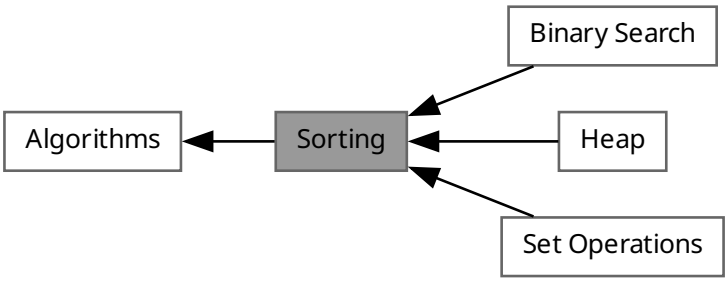
Returns

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

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

3.1.5 Sorting

Collaboration diagram for Sorting:



Topics

- Binary Search
- Heap
- Set Operations

Functions

- template<typename _Tp>
  constexpr const _Tp & std::clamp (const _Tp &__val, const _Tp &__lo, const _Tp &__hi)
- template<typename _Tp, typename _Compare>
  constexpr const _Tp & std::clamp (const _Tp &__val, const _Tp &__lo, const _Tp &__hi, _Compare __comp)
- template<typename _BidirectionalIterator>
  void std::inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __↩ last)
- template<typename _BidirectionalIterator, typename _Compare>
  void std::inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __↩ last, _Compare __comp)
- template<typename _ForwardIterator>
  constexpr bool std::is_sorted (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _Compare>
  constexpr bool std::is_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)
- template<typename _ForwardIterator>
  constexpr _ForwardIterator std::is_sorted_until (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _Compare>
  constexpr _ForwardIterator std::is_sorted_until (_ForwardIterator __first, _ForwardIterator __last, _Compare _↩ _comp)
- template<typename _II1, typename _II2>
  constexpr bool std::lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)

- template<typename _Il1, typename _Il2, typename _Compare>
  constexpr bool std::lexicographical_compare (_Il1 __first1, _Il1 __last1, _Il2 __first2, _Il2 __last2, _Compare __comp)
- template<typename _InputIter1, typename _InputIter2, typename _Comp>
  constexpr auto std::lexicographical_compare_three_way (_InputIter1 __first1, _InputIter1 __last1, _InputIter2 _↩ _first2, _InputIter2 __last2, _Comp __comp) -> decltype(__comp(*__first1, *__first2))
- template<typename _Tp>
  constexpr const _Tp & std::max (const _Tp &__a, const _Tp &__b)
- template<typename _Tp, typename _Compare>
  constexpr const _Tp & std::max (const _Tp &__a, const _Tp &__b, _Compare __comp)
- template<typename _ForwardIterator>
  constexpr _ForwardIterator std::max_element (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _Compare>
  constexpr _ForwardIterator std::max_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __↩ comp)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>
  constexpr _OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _Tp>
  constexpr const _Tp & std::min (const _Tp &__a, const _Tp &__b)
- template<typename _Tp, typename _Compare>
  constexpr const _Tp & std::min (const _Tp &__a, const _Tp &__b, _Compare __comp)
- template<typename _ForwardIterator>
  constexpr _ForwardIterator std::min_element (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _Compare>
  constexpr _ForwardIterator std::min_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __↩ comp)
- template<typename _Tp>
  constexpr pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b)
- template<typename _Tp, typename _Compare>
  constexpr pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b, _Compare __comp)
- template<typename _ForwardIterator>
  constexpr pair< _ForwardIterator, _ForwardIterator > std::minmax_element (_ForwardIterator __first, _↩ ForwardIterator __last)
- template<typename _ForwardIterator, typename _Compare>
  constexpr pair< _ForwardIterator, _ForwardIterator > std::minmax_element (_ForwardIterator __first, _↩ ForwardIterator __last, _Compare __comp)
- template<typename _BidirectionalIterator>
  constexpr bool std::next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)
- template<typename _BidirectionalIterator, typename _Compare>
  constexpr bool std::next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __↩ comp)
- template<typename _RandomAccessIterator>
  constexpr void std::nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _Random↩ AccessIterator __last)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void std::nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _Random↩ AccessIterator __last, _Compare __comp)

- template<typename _RandomAccessIterator>
  constexpr void std::partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _Random↩
  AccessIterator __last)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void std::partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _Random↩
  AccessIterator __last, _Compare __comp)
- template<typename _InputIterator, typename _RandomAccessIterator>
  constexpr _RandomAccessIterator std::partial_sort_copy (_InputIterator __first, _InputIterator __last, _Random↩
  AccessIterator __result_first, _RandomAccessIterator __result_last)
- template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare>
  constexpr _RandomAccessIterator std::partial_sort_copy (_InputIterator __first, _InputIterator __last, _Random↩
  AccessIterator __result_first, _RandomAccessIterator __result_last, _Compare __comp)
- template<typename _BidirectionalIterator>
  constexpr bool std::prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)
- template<typename _BidirectionalIterator, typename _Compare>
  constexpr bool std::prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __↩
  comp)
- template<typename _RandomAccessIterator>
  constexpr void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _RandomAccessIterator>
  void std::stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _Compare>
  void std::stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)

3.1.5.1 Detailed Description

3.1.5.2 Function Documentation

clamp() `[1/2]`

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

Returns the value clamped between lo and hi.

Parameters

__val	A value of arbitrary type.
__lo	A lower limit of arbitrary type.
__hi	An upper limit of arbitrary type.

Return values

`__lo`	if `__val < __lo`
`__hi`	if `__hi < __val`
`__val`	otherwise.

Precondition

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

References max(), and min().

clamp() [2/2]

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

Returns the value clamped between lo and hi.

Parameters

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

Return values

`__lo`	if `__comp(__val, __lo)`
`__hi`	if `__comp(__hi, __val)`
`__val`	otherwise.

Precondition

`__comp(__hi, __lo)` is false.

References max(), and min().

inplace_merge() [1/2]

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

Merges two sorted ranges in place.

Parameters

__first	An iterator.
__middle	Another iterator.
__last	Another iterator.

Returns

Nothing.

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

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

inplace_merge() [2/2]

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

Merges two sorted ranges in place.

Parameters

__first	An iterator.
__middle	Another iterator.
__last	Another iterator.
__comp	A functor to use for comparisons.

Returns

Nothing.

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

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

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

is_sorted() [1/2]

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

Determines whether the elements of a sequence are sorted.

Parameters

__first	An iterator.
__last	Another iterator.

Returns

True if the elements are sorted, false otherwise.

is_sorted() [2/2]

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

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

Parameters

__first	An iterator.
__last	Another iterator.
__comp	A comparison functor.

Returns

True if the elements are sorted, false otherwise.

is_sorted_until() [1/2]

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

Determines the end of a sorted sequence.

Parameters

__first	An iterator.
__last	Another iterator.

Returns

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

is_sorted_until() [2/2]

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

Determines the end of a sorted sequence using comparison functor.

Parameters

__first	An iterator.
__last	Another iterator.
__comp	A comparison functor.

Returns

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

lexicographical_compare() [1/2]

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

Performs **dictionary** comparison on ranges.

Parameters

__first1	An input iterator.
__last1	An input iterator.
__first2	An input iterator.
__last2	An input iterator.

Returns

A boolean true or false.

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

lexicographical_compare() [2/2]

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

Performs **dictionary** comparison on ranges.

Parameters

__first1	An input iterator.
__last1	An input iterator.
__first2	An input iterator.
__last2	An input iterator.
__comp	A comparison functor.

Returns

A boolean true or false.

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

lexicographical_compare_three_way()

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

Performs dictionary comparison on ranges.

Parameters

__first1	An input iterator.
__last1	An input iterator.
__first2	An input iterator.
__last2	An input iterator.
__comp	A comparison functor.

Returns

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

Referenced by operator<=>().

max() [1/2]

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

This does what you think it does.

Parameters

_↩ _a	A thing of arbitrary type.
_↩ _b	Another thing of arbitrary type.

Returns

The greater of the parameters.

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

Referenced by __gnu_parallel::__find_template(), __gnu_parallel::__parallel_nth_element(), __gnu_parallel::__parallel_partial_sum_linea __gnu_parallel::__parallel_partition(), __gnu_parallel::__parallel_random_shuffle_drs(), __gnu_parallel::__qsb_conquer(), __gnu_parallel::__search_template(), __gnu_parallel::__sequential_random_shuffle(), std::_Deque_base< _Tp, _Alloc >::_M_initialize_ std::deque< _Tp, _Alloc >::_M_reallocate_map(), clamp(), clamp(), generate_canonical(), __gnu_parallel::multiseq_partition(), __gnu_parallel::multiseq_selection(), std::shuffle_order_engine< _RandomNumberEngine, __k >::operator()(), std::shuffle_order_engine< minstd_rand0, 256 >::operator()(), std::subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 >::operator()(and std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow().

max() [2/2]

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

This does what you think it does.

Parameters

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

Returns

The greater of the parameters.

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

max_element() [1/2]

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

Return the maximum element in a range.

Parameters

__first	Start of range.
__last	End of range.

Returns

Iterator referencing the first instance of the largest value.

max_element() [2/2]

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

Return the maximum element in a range using comparison functor.

Parameters

__first	Start of range.
__last	End of range.
__comp	Comparison functor.

Returns

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

merge() [1/2]

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

Merges two sorted ranges.

Parameters

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

Returns

An output iterator equal to __result + (__last1 - __first1)

- (__last2 - __first2).

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

merge() [2/2]

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

Merges two sorted ranges.

Parameters

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

Returns

An output iterator equal to __result + (__last1 - __first1)

- (__last2 - __first2).

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

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

min() [1/2]

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

This does what you think it does.

Parameters

_↩ _a	A thing of arbitrary type.
_↩ _b	Another thing of arbitrary type.

Returns

The lesser of the parameters.

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

Referenced by __gnu_parallel::__find_template(), __gnu_parallel::__find_template(), __gnu_parallel::__for_each_template_random_acce __gnu_parallel::__parallel_partial_sum_linear(), __gnu_parallel::__parallel_random_shuffle_drs(), __gnu_parallel::__parallel_sort_qs_div __gnu_parallel::__qsb_conquer(), __sample(), __gnu_parallel::__search_template(), __gnu_parallel::__sequential_multiway_merge(), __gnu_parallel::__sequential_random_shuffle(), clamp(), clamp(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compar __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare(), __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::__versa_string< char >::compare(), std::basic_string< _CharT >::compare(), std::basic_string< _CharT >::compare(), std::basic_string< _CharT >::compare(), std::basic_string< _CharT >::compare(), std::basic_string< _CharT >::compare(), generate_canonical(), std::basic_string< _CharT >::max_size(), __gnu_parallel::multiseq_parti __gnu_parallel::multiseq_selection(), std::shuffle_order_engine< _RandomNumberEngine, __k >::operator()(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), __gnu_parallel::parallel_multiway_merge(), std::basic_istream< _CharT, _Tra __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rfind(), std::basic_string< _CharT, _Traits, _Alloc >::rfind(), std::basic_streambuf< _CharT, _Traits >::xsgetn(), and std::basic_streambuf< _CharT, _Traits >::xsputn().

min() [2/2]

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

This does what you think it does.

Parameters

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

Returns

The lesser of the parameters.

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

min_element() [1/2]

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

Return the minimum element in a range.

Parameters

__first	Start of range.
__last	End of range.

Returns

Iterator referencing the first instance of the smallest value.

min_element() [2/2]

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

Return the minimum element in a range using comparison functor.

Parameters

__first	Start of range.
__last	End of range.
__comp	Comparison functor.

Returns

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

minmax() [1/2]

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

Determines min and max at once as an ordered pair.

Parameters

_↩ _a	A thing of arbitrary type.
_↩ _b	Another thing of arbitrary type.

Returns

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

minmax() [2/2]

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

Determines min and max at once as an ordered pair.

Parameters

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

Returns

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

minmax_element() [1/2]

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

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

Parameters

__first	Start of range.
__last	End of range.

Returns

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

minmax_element() [2/2]

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

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

Parameters

__first	Start of range.
__last	End of range.
__comp	Comparison functor.

Returns

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

next_permutation() [1/2]

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

Permute range into the next *dictionary* ordering.

Parameters

__first	Start of range.
__last	End of range.

Returns

False if wrapped to first permutation, true otherwise.

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

next_permutation() [2/2]

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

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

Parameters

__first	Start of range.
__last	End of range.
__comp	A comparison functor.

Returns

False if wrapped to first permutation, true otherwise.

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

nth_element() [1/2]

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

Sort a sequence just enough to find a particular position.

Parameters

__first	An iterator.
__nth	Another iterator.
__last	Another iterator.

Returns

Nothing.

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

References __lg().

nth_element() [2/2]

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

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

Parameters

__first	An iterator.
__nth	Another iterator.
__last	Another iterator.
__comp	A comparison functor.

Returns

Nothing.

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

References __lg().

partial_sort() [1/2]

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

Sort the smallest elements of a sequence.

Parameters

__first	An iterator.
__middle	Another iterator.
__last	Another iterator.

Returns

Nothing.

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

partial_sort() [2/2]

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

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

Parameters

__first	An iterator.
__middle	Another iterator.
__last	Another iterator.
__comp	A comparison functor.

Returns

Nothing.

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

partial_sort_copy() [1/2]

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

Copy the smallest elements of a sequence.

Parameters

__first	An iterator.
__last	Another iterator.
__result_first	A random-access iterator.
__result_last	Another random-access iterator.

Returns

An iterator indicating the end of the resulting sequence.

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

partial_sort_copy() [2/2]

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

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

Parameters

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

Returns

An iterator indicating the end of the resulting sequence.

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

prev_permutation() [1/2]

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

Permute range into the previous *dictionary* ordering.

Parameters

__first	Start of range.
__last	End of range.

Returns

False if wrapped to last permutation, true otherwise.

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

prev_permutation() [2/2]

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

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

Parameters

__first	Start of range.
__last	End of range.
__comp	A comparison functor.

Returns

False if wrapped to last permutation, true otherwise.

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

sort() [1/2]

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

Sort the elements of a sequence.

Parameters

__first	An iterator.
__last	Another iterator.

Returns

Nothing.

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

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

sort() [2/2]

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

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

Parameters

__first	An iterator.
__last	Another iterator.
__comp	A comparison functor.

Returns

Nothing.

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

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

stable_sort() [1/2]

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

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

Parameters

__first	An iterator.
__last	Another iterator.

Returns

Nothing.

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

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

stable_sort() [2/2]

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

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

Parameters

__first	An iterator.
__last	Another iterator.
__comp	A comparison functor.

Returns

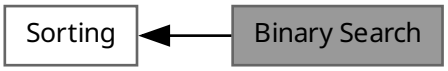
Nothing.

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

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

3.1.5.3 Binary Search

Collaboration diagram for Binary Search:



Functions

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

- template<typename _ForwardIterator, typename _Tp>
  constexpr _ForwardIterator std::upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__↩
  val)
- template<typename _ForwardIterator, typename _Tp, typename _Compare>
  constexpr _ForwardIterator std::upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__↩
  val, _Compare __comp)

3.1.5.3.1 Detailed Description

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

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

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

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

3.1.5.3.2 Function Documentation

binary_search() [1/2]

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

Determines whether an element exists in a range.

Parameters

__first	An iterator.
__last	Another iterator.
__val	The search term.

Returns

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

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

binary_search() [2/2]

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

Determines whether an element exists in a range.

Parameters

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

Returns

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

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

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

equal_range() [1/2]

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

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

Parameters

__first	An iterator.
__last	Another iterator.
__val	The search term.

Returns

An pair of iterators defining the subrange.

This is equivalent to

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

but does not actually call those functions.

equal_range() [2/2]

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

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

Parameters

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

Returns

An pair of iterators defining the subrange.

This is equivalent to

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

but does not actually call those functions.

lower_bound() [1/2]

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

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

Parameters

__first	An iterator.
__last	Another iterator.
__val	The search term.

Returns

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

lower_bound() [2/2]

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

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

Parameters

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

Returns

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

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

upper_bound() [1/2]

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

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

Parameters

__first	An iterator.
__last	Another iterator.
__val	The search term.

Returns

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

upper_bound() [2/2]

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

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

Parameters

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

Returns

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

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

3.1.5.4 Heap

Collaboration diagram for Heap:

Functions

- template<typename _RandomAccessIterator>
  constexpr bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _RandomAccessIterator>
  constexpr _RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr _RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _RandomAccessIterator>
  constexpr void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare _↩ _comp)
- template<typename _RandomAccessIterator>
  constexpr void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)

- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __↩
  comp)
- template<typename _RandomAccessIterator>
  constexpr void std::push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void std::push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __↩
  comp)
- template<typename _RandomAccessIterator>
  constexpr void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __↩
  comp)

3.1.5.4.1 Detailed Description

3.1.5.4.2 Function Documentation

is_heap() [1/2]

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

Determines whether a range is a heap.

Parameters

__first	Start of range.
__last	End of range.

Returns

True if range is a heap, false otherwise.

is_heap() [2/2]

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

Determines whether a range is a heap using comparison functor.

Parameters

__first	Start of range.
__last	End of range.
__comp	Comparison functor to use.

Returns

True if range is a heap, false otherwise.

References distance().

is_heap_until() [1/2]

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

Search the end of a heap.

Parameters

__first	Start of range.
__last	End of range.

Returns

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

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

References distance().

is_heap_until() [2/2]

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

Search the end of a heap using comparison functor.

Parameters

__first	Start of range.
__last	End of range.
__comp	Comparison functor to use.

Returns

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

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

References distance().

make_heap() [1/2]

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

Construct a heap over a range.

Parameters

__first	Start of heap.
__last	End of heap.

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

make_heap() [2/2]

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

Construct a heap over a range using comparison functor.

Parameters

__first	Start of heap.
__last	End of heap.
__comp	Comparison functor to use.

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

pop_heap() [1/2]

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

Pop an element off a heap.

Parameters

__first	Start of heap.
__last	End of heap.

Precondition

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

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

pop_heap() [2/2]

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

Pop an element off a heap using comparison functor.

Parameters

__first	Start of heap.
__last	End of heap.
__comp	Comparison functor to use.

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

push_heap() [1/2]

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

Push an element onto a heap.

Parameters

__first	Start of heap.
__last	End of heap + element.

This operation pushes the element at last-1 onto the valid heap over the range [__first,__last-1). After completion, [__first,__last) is a valid heap.

push_heap() [2/2]

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

Push an element onto a heap using comparison functor.

Parameters

__first	Start of heap.
__last	End of heap + element.
__comp	Comparison functor.

This operation pushes the element at __last-1 onto the valid heap over the range [__first,__last-1). After completion, [__first,__last) is a valid heap. Compare operations are performed using comp.

sort_heap() [1/2]

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

Sort a heap.

Parameters

__first	Start of heap.
__last	End of heap.

This operation sorts the valid heap in the range [__first,__last).

sort_heap() [2/2]

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

Sort a heap using comparison functor.

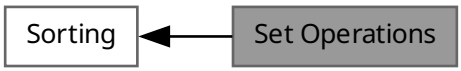
Parameters

__first	Start of heap.
__last	End of heap.
__comp	Comparison functor to use.

This operation sorts the valid heap in the range [__first,__last). Comparisons are made using __comp.

3.1.5.5 Set Operations

Collaboration diagram for Set Operations:



Functions

- template<typename _InputIterator1, typename _InputIterator2>
  constexpr bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Input↩ Iterator2 __last2)
- template<typename _InputIterator1, typename _InputIterator2, typename _Compare>
  constexpr bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Input↩ Iterator2 __last2, _Compare __comp)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>
  constexpr _OutputIterator std::set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 _↩ _first2, _InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator std::set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 _↩ _first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>
  constexpr _OutputIterator std::set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator std::set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>
  constexpr _OutputIterator std::set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, _↩ InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator std::set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, _↩ InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)

- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>
  constexpr _OutputIterator std::set_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator std::set_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)

3.1.5.5.1 Detailed Description

These algorithms are common set operations performed on sequences that are already sorted. The number of comparisons will be linear.

3.1.5.5.2 Function Documentation

includes() [1/2]

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

Determines whether all elements of a sequence exists in a range.

Parameters

__first1	Start of search range.
__last1	End of search range.
__first2	Start of sequence
__last2	End of sequence.

Returns

True if each element in [__first2,__last2) is contained in order within [__first1,__last1). False otherwise.

This operation expects both [__first1,__last1) and [__first2,__last2) to be sorted. Searches for the presence of each element in [__first2,__last2) within [__first1,__last1). The iterators over each range only move forward, so this is a linear algorithm. If an element in [__first2,__last2) is not found before the search iterator reaches __last2, false is returned.

includes() [2/2]

```
template<typename _InputIterator1, typename _InputIterator2, typename _Compare>
bool std::includes (
            _InputIterator1 __first1,
            _InputIterator1 __last1,
            _InputIterator2 __first2,
            _InputIterator2 __last2,
            _Compare __comp)  [inline], [nodiscard], [constexpr]
```

Determines whether all elements of a sequence exists in a range using comparison.

Parameters

__first1	Start of search range.
__last1	End of search range.
__first2	Start of sequence
__last2	End of sequence.
__comp	Comparison function to use.

Returns

True if each element in [__first2,__last2) is contained in order within [__first1,__last1) according to comp. False otherwise.

This operation expects both [__first1,__last1) and [__first2,__last2) to be sorted. Searches for the presence of each element in [__first2,__last2) within [__first1,__last1), using comp to decide. The iterators over each range only move forward, so this is a linear algorithm. If an element in [__first2,__last2) is not found before the search iterator reaches `__last2`, false is returned.

set_difference() [1/2]

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

Return the difference of two sorted ranges.

Parameters

__first1	Start of first range.
__last1	End of first range.
__first2	Start of second range.
__last2	End of second range.
__result	Start of output range.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in the first range but not the second in order to the output range. Iterators increment for each range. When the current element of the first range is less than the second, that element is copied and the iterator advances. If the current element of the second range is less, the iterator advances, but no element is copied. If an element is contained in both ranges, no elements are copied and both ranges advance. The output range may not overlap either input range.

set_difference() [2/2]

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

Return the difference of two sorted ranges using comparison functor.

Parameters

__first1	Start of first range.
__last1	End of first range.
__first2	Start of second range.
__last2	End of second range.
__result	Start of output range.
__comp	The comparison functor.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in the first range but not the second in order to the output range. Iterators increment for each range. When the current element of the first range is less than the second according to __comp, that element is copied and the iterator advances. If the current element of the second range is less, no element is copied and the iterator advances. If an element is contained in both ranges according to __comp, no elements are copied and both ranges advance. The output range may not overlap either input range.

set_intersection() [1/2]

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

Return the intersection of two sorted ranges.

Parameters

__first1	Start of first range.
__last1	End of first range.

__first2	Start of second range.
__last2	End of second range.
__result	Start of output range.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in both ranges in order to the output range. Iterators increment for each range. When the current element of one range is less than the other, that iterator advances. If an element is contained in both ranges, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

set_intersection() [2/2]

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

Return the intersection of two sorted ranges using comparison functor.

Parameters

__first1	Start of first range.
__last1	End of first range.
__first2	Start of second range.
__last2	End of second range.
__result	Start of output range.
__comp	The comparison functor.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in both ranges in order to the output range. Iterators increment for each range. When the current element of one range is less than the other according to __comp, that iterator advances. If an element is contained in both ranges according to __comp, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

set_symmetric_difference() [1/2]

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

Return the symmetric difference of two sorted ranges.

Parameters

__first1	Start of first range.
__last1	End of first range.
__first2	Start of second range.
__last2	End of second range.
__result	Start of output range.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in one range but not the other in order to the output range. Iterators increment for each range. When the current element of one range is less than the other, that element is copied and the iterator advances. If an element is contained in both ranges, no elements are copied and both ranges advance. The output range may not overlap either input range.

set_symmetric_difference() [2/2]

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

Return the symmetric difference of two sorted ranges using comparison functor.

Parameters

__first1	Start of first range.
__last1	End of first range.
__first2	Start of second range.
__last2	End of second range.
__result	Start of output range.
__comp	The comparison functor.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in one range but not the other in order to the output range. Iterators increment for each range. When the current element of one range is less than the other according to `comp`, that element is copied and the iterator advances. If an element is contained in both ranges according to `__comp`, no elements are copied and both ranges advance. The output range may not overlap either input range.

set_union() [1/2]

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

Return the union of two sorted ranges.

Parameters

__first1	Start of first range.
__last1	End of first range.
__first2	Start of second range.
__last2	End of second range.
__result	Start of output range.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in each range in order to the output range. Iterators increment for each range. When the current element of one range is less than the other, that element is copied and the iterator advanced. If an element is contained in both ranges, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

set_union() [2/2]

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

Return the union of two sorted ranges using a comparison functor.

Parameters

__first1	Start of first range.
__last1	End of first range.
__first2	Start of second range.
__last2	End of second range.
__result	Start of output range.
__comp	The comparison functor.

Returns

End of the output range.

This operation iterates over both ranges, copying elements present in each range in order to the output range. Iterators increment for each range. When the current element of one range is less than the other according to `__comp`, that element is copied and the iterator advanced. If an equivalent element according to `__comp` is contained in both ranges, the element from the first range is copied and both ranges advance. The output range may not overlap either input range.

3.2 Atomics

Classes

- struct std::atomic< _Tp >
- struct std::atomic< _Tp * >
- struct std::atomic_flag

Macros

- #define ATOMIC_BOOL_LOCK_FREE
- #define **ATOMIC_CHAR16_T_LOCK_FREE**
- #define **ATOMIC_CHAR32_T_LOCK_FREE**
- #define **ATOMIC_CHAR_LOCK_FREE**
- #define **ATOMIC_FLAG_INIT**
- #define **ATOMIC_INT_LOCK_FREE**
- #define **ATOMIC_LLONG_LOCK_FREE**
- #define **ATOMIC_LONG_LOCK_FREE**
- #define **ATOMIC_POINTER_LOCK_FREE**
- #define **ATOMIC_SHORT_LOCK_FREE**
- #define **ATOMIC_VAR_INIT**(_VI)
- #define **ATOMIC_WCHAR_T_LOCK_FREE**

Typedefs

- typedef unsigned char **std::__atomic_flag_data_type**
- typedef atomic< bool > std::atomic_bool
- typedef atomic< char > std::atomic_char
- typedef atomic< char16_t > std::atomic_char16_t
- typedef atomic< char32_t > std::atomic_char32_t
- typedef atomic< int > std::atomic_int
- typedef atomic< int16_t > std::atomic_int16_t
- typedef atomic< int32_t > std::atomic_int32_t
- typedef atomic< int64_t > std::atomic_int64_t
- typedef atomic< int8_t > std::atomic_int8_t
- typedef atomic< int_fast16_t > std::atomic_int_fast16_t
- typedef atomic< int_fast32_t > std::atomic_int_fast32_t
- typedef atomic< int_fast64_t > std::atomic_int_fast64_t
- typedef atomic< int_fast8_t > std::atomic_int_fast8_t
- typedef atomic< int_least16_t > std::atomic_int_least16_t
- typedef atomic< int_least32_t > std::atomic_int_least32_t
- typedef atomic< int_least64_t > std::atomic_int_least64_t
- typedef atomic< int_least8_t > std::atomic_int_least8_t
- typedef atomic< intmax_t > std::atomic_intmax_t
- typedef atomic< intptr_t > std::atomic_intptr_t
- typedef atomic< long long > std::atomic_llong
- typedef atomic< long > std::atomic_long
- typedef atomic< ptrdiff_t > std::atomic_ptrdiff_t
- typedef atomic< signed char > std::atomic_schar
- typedef atomic< short > std::atomic_short
- typedef atomic< size_t > std::atomic_size_t
- typedef atomic< unsigned char > std::atomic_uchar
- typedef atomic< unsigned int > std::atomic_uint
- typedef atomic< uint16_t > std::atomic_uint16_t
- typedef atomic< uint32_t > std::atomic_uint32_t
- typedef atomic< uint64_t > std::atomic_uint64_t
- typedef atomic< uint8_t > std::atomic_uint8_t
- typedef atomic< uint_fast16_t > std::atomic_uint_fast16_t
- typedef atomic< uint_fast32_t > std::atomic_uint_fast32_t
- typedef atomic< uint_fast64_t > std::atomic_uint_fast64_t
- typedef atomic< uint_fast8_t > std::atomic_uint_fast8_t
- typedef atomic< uint_least16_t > std::atomic_uint_least16_t
- typedef atomic< uint_least32_t > std::atomic_uint_least32_t
- typedef atomic< uint_least64_t > std::atomic_uint_least64_t
- typedef atomic< uint_least8_t > std::atomic_uint_least8_t
- typedef atomic< uintmax_t > std::atomic_uintmax_t
- typedef atomic< uintptr_t > std::atomic_uintptr_t
- typedef atomic< unsigned long long > std::atomic_ullong
- typedef atomic< unsigned long > std::atomic_ulong
- typedef atomic< unsigned short > std::atomic_ushort
- typedef atomic< wchar_t > std::atomic_wchar_t

Enumerations

- enum class std::memory_order : int {
  relaxed , **consume** , **acquire** , **release** ,
  acq_rel , **seq_cst** }

Functions

- template<typename _ITp>
  bool **std::atomic_compare_exchange_strong** (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2) noexcept
- template<typename _ITp>
  bool **std::atomic_compare_exchange_strong** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2) noexcept
- template<typename _ITp>
  bool **std::atomic_compare_exchange_strong_explicit** (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2, memory_order __m1, memory_order __m2) noexcept
- template<typename _ITp>
  bool **std::atomic_compare_exchange_strong_explicit** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2, memory_order __m1, memory_order __m2) noexcept
- template<typename _ITp>
  bool **std::atomic_compare_exchange_weak** (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2) noexcept
- template<typename _ITp>
  bool **std::atomic_compare_exchange_weak** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2) noexcept
- template<typename _ITp>
  bool **std::atomic_compare_exchange_weak_explicit** (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2, memory_order __m1, memory_order __m2) noexcept
- template<typename _ITp>
  bool **std::atomic_compare_exchange_weak_explicit** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2, memory_order __m1, memory_order __m2) noexcept
- template<typename _ITp>
  _ITp **std::atomic_exchange** (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **std::atomic_exchange** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **std::atomic_exchange_explicit** (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **std::atomic_exchange_explicit** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_add** (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_add** (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_add_explicit** (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_add_explicit** (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order __m) noexcept

- template<typename _ITp>
  _ITp **std::atomic_fetch_and** (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_and** (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_and_explicit** (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_and_explicit** (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_or** (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_or** (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_or_explicit** (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_or_explicit** (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_sub** (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_sub** (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_sub_explicit** (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_sub_explicit** (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_xor** (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_xor** (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_xor_explicit** (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_xor_explicit** (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- void **std::atomic_flag_clear** (atomic_flag *__a) noexcept
- void **std::atomic_flag_clear** (volatile atomic_flag *__a) noexcept
- void **std::atomic_flag_clear_explicit** (atomic_flag *__a, memory_order __m) noexcept
- void **std::atomic_flag_clear_explicit** (volatile atomic_flag *__a, memory_order __m) noexcept
- bool **std::atomic_flag_test_and_set** (atomic_flag *__a) noexcept
- bool **std::atomic_flag_test_and_set** (volatile atomic_flag *__a) noexcept
- bool **std::atomic_flag_test_and_set_explicit** (atomic_flag *__a, memory_order __m) noexcept
- bool **std::atomic_flag_test_and_set_explicit** (volatile atomic_flag *__a, memory_order __m) noexcept
- template<typename _ITp>
  void **std::atomic_init** (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  void **std::atomic_init** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept

- template<typename _ITp>
  bool **std::atomic_is_lock_free** (const atomic< _ITp > *__a) noexcept
- template<typename _ITp>
  bool **std::atomic_is_lock_free** (const volatile atomic< _ITp > *__a) noexcept
- template<typename _ITp>
  _ITp **std::atomic_load** (const atomic< _ITp > *__a) noexcept
- template<typename _ITp>
  _ITp **std::atomic_load** (const volatile atomic< _ITp > *__a) noexcept
- template<typename _ITp>
  _ITp **std::atomic_load_explicit** (const atomic< _ITp > *__a, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **std::atomic_load_explicit** (const volatile atomic< _ITp > *__a, memory_order __m) noexcept
- void **std::atomic_signal_fence** (memory_order __m) noexcept
- template<typename _ITp>
  void **std::atomic_store** (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  void **std::atomic_store** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  void **std::atomic_store_explicit** (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  void **std::atomic_store_explicit** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- void **std::atomic_thread_fence** (memory_order __m) noexcept
- template<typename _Tp>
  _Tp std::kill_dependency (_Tp __y) noexcept
- constexpr memory_order **std::operator&** (memory_order __m, __memory_order_modifier __mod) noexcept
- constexpr memory_order **std::operator|** (memory_order __m, __memory_order_modifier __mod) noexcept

Variables

- constexpr memory_order **std::memory_order_acq_rel**
- constexpr memory_order **std::memory_order_acquire**
- constexpr memory_order **std::memory_order_consume**
- constexpr memory_order **std::memory_order_relaxed**
- constexpr memory_order **std::memory_order_release**
- constexpr memory_order **std::memory_order_seq_cst**

3.2.1 Detailed Description

Components for performing atomic operations.

3.2.2 Macro Definition Documentation

ATOMIC_BOOL_LOCK_FREE

```
#define ATOMIC_BOOL_LOCK_FREE
```

Lock-free property.

0 indicates that the types are never lock-free. 1 indicates that the types are sometimes lock-free. 2 indicates that the types are always lock-free.

3.2.3 Typedef Documentation

atomic_bool

```
typedef atomic<bool> std::atomic_bool
```

atomic_bool

atomic_char

```
typedef atomic<char> std::atomic_char
```

atomic_char

atomic_char16_t

```
typedef atomic<char16_t> std::atomic_char16_t
```

atomic_char16_t

atomic_char32_t

```
typedef atomic<char32_t> std::atomic_char32_t
```

atomic_char32_t

atomic_int

```
typedef atomic<int> std::atomic_int
```

atomic_int

atomic_int16_t

```
typedef atomic<int16_t> std::atomic_int16_t
```

atomic_int16_t

atomic_int32_t

```
typedef atomic<int32_t> std::atomic_int32_t
```

atomic_int32_t

atomic_int64_t

```
typedef atomic<int64_t> std::atomic_int64_t
```

atomic_int64_t

atomic_int8_t

```
typedef atomic<int8_t> std::atomic_int8_t
```

atomic_int8_t

atomic_int_fast16_t

```
typedef atomic<int_fast16_t> std::atomic_int_fast16_t
```

atomic_int_fast16_t

atomic_int_fast32_t

```
typedef atomic<int_fast32_t> std::atomic_int_fast32_t
```

atomic_int_fast32_t

atomic_int_fast64_t

```
typedef atomic<int_fast64_t> std::atomic_int_fast64_t
```

atomic_int_fast64_t

atomic_int_fast8_t

```
typedef atomic<int_fast8_t> std::atomic_int_fast8_t
```

atomic_int_fast8_t

atomic_int_least16_t

```
typedef atomic<int_least16_t> std::atomic_int_least16_t
```

atomic_int_least16_t

atomic_int_least32_t

```
typedef atomic<int_least32_t> std::atomic_int_least32_t
```

atomic_int_least32_t

atomic_int_least64_t

```
typedef atomic<int_least64_t> std::atomic_int_least64_t
```

atomic_int_least64_t

atomic_int_least8_t

```
typedef atomic<int_least8_t> std::atomic_int_least8_t
```

atomic_int_least8_t

atomic_intmax_t

```
typedef atomic<intmax_t> std::atomic_intmax_t
```

atomic_intmax_t

atomic_intptr_t

```
typedef atomic<intptr_t> std::atomic_intptr_t
```

atomic_intptr_t

atomic_llong

```
typedef atomic<long long> std::atomic_llong
```

atomic_llong

atomic_long

```
typedef atomic<long> std::atomic_long
```

atomic_long

atomic_ptrdiff_t

```
typedef atomic<ptrdiff_t> std::atomic_ptrdiff_t
```

atomic_ptrdiff_t

atomic_schar

```
typedef atomic<signed char> std::atomic_schar
```

atomic_schar

atomic_short

```
typedef atomic<short> std::atomic_short
```

atomic_short

atomic_size_t

```
typedef atomic<size_t> std::atomic_size_t
```

atomic_size_t

atomic_uchar

```
typedef atomic<unsigned char> std::atomic_uchar
```

atomic_uchar

atomic_uint

```
typedef atomic<unsigned int> std::atomic_uint
```

atomic_uint

atomic_uint16_t

```
typedef atomic<uint16_t> std::atomic_uint16_t
```

atomic_uint16_t

atomic_uint32_t

```
typedef atomic<uint32_t> std::atomic_uint32_t
```

atomic_uint32_t

atomic_uint64_t

```
typedef atomic<uint64_t> std::atomic_uint64_t
```

atomic_uint64_t

atomic_uint8_t

```
typedef atomic<uint8_t> std::atomic_uint8_t
```

atomic_uint8_t

atomic_uint_fast16_t

```
typedef atomic<uint_fast16_t> std::atomic_uint_fast16_t
```

atomic_uint_fast16_t

atomic_uint_fast32_t

```
typedef atomic<uint_fast32_t> std::atomic_uint_fast32_t
```

atomic_uint_fast32_t

atomic_uint_fast64_t

```
typedef atomic<uint_fast64_t> std::atomic_uint_fast64_t
```

atomic_uint_fast64_t

atomic_uint_fast8_t

```
typedef atomic<uint_fast8_t> std::atomic_uint_fast8_t
```

atomic_uint_fast8_t

atomic_uint_least16_t

```
typedef atomic<uint_least16_t> std::atomic_uint_least16_t
```

atomic_uint_least16_t

atomic_uint_least32_t

```
typedef atomic<uint_least32_t> std::atomic_uint_least32_t
```

atomic_uint_least32_t

atomic_uint_least64_t

```
typedef atomic<uint_least64_t> std::atomic_uint_least64_t
```

atomic_uint_least64_t

atomic_uint_least8_t

```
typedef atomic<uint_least8_t> std::atomic_uint_least8_t
```

atomic_uint_least8_t

atomic_uintmax_t

```
typedef atomic<uintmax_t> std::atomic_uintmax_t
```

atomic_uintmax_t

atomic_uintptr_t

```
typedef atomic<uintptr_t> std::atomic_uintptr_t
```

atomic_uintptr_t

atomic_ullong

```
typedef atomic<unsigned long long> std::atomic_ullong
```

atomic_ullong

atomic_ulong

```
typedef atomic<unsigned long> std::atomic_ulong
```

atomic_ulong

atomic_ushort

```
typedef atomic<unsigned short> std::atomic_ushort
```

atomic_ushort

atomic_wchar_t

```
typedef atomic<wchar_t> std::atomic_wchar_t
```

atomic_wchar_t

3.2.4 Enumeration Type Documentation

memory_order

```
enum class std::memory_order :  int  [strong]
```

Enumeration for memory_order.

3.2.5 Function Documentation

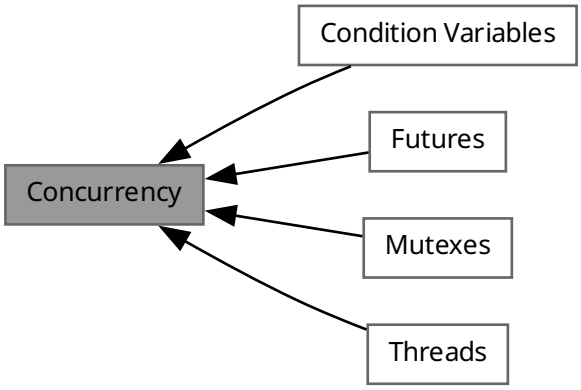
kill_dependency()

```
template<typename _Tp>
_Tp std::kill_dependency (
             _Tp __y)  [inline], [noexcept]
```

kill_dependency

3.3 Concurrency

Collaboration diagram for Concurrency:



Topics

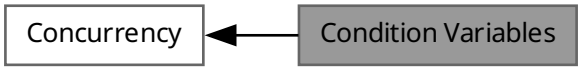
- Condition Variables
- Futures
- Mutexes
- Threads

3.3.1 Detailed Description

Components for concurrent operations, including threads, mutexes, and condition variables.

3.3.2 Condition Variables

Collaboration diagram for Condition Variables:



Classes

- class std::condition_variable
- class std::condition_variable_any

Enumerations

- enum class std::cv_status { **no_timeout** , **timeout** }

Functions

- void **std::notify_all_at_thread_exit** (condition_variable &, unique_lock< mutex >)

3.3.2.1 Detailed Description

Classes for condition_variable support.

3.3.2.2 Enumeration Type Documentation

cv_status

```
enum class std::cv_status  [strong]
```

cv_status

3.3.3 Futures

Collaboration diagram for Futures:

Classes

- class std::__basic_future< _Res >
- class std::future< _Res >
- class std::future< _Res & >
- class std::future< void >
- class std::future_error
- struct std::is_error_code_enum< future_errc >
- class std::packaged_task< _Res(_ArgTypes...)>
- class std::promise< _Res >
- class std::promise< _Res & >
- class std::promise< void >
- class std::shared_future< _Res >
- class std::shared_future< _Res & >
- class std::shared_future< void >

Enumerations

- enum class std::future_errc { **future_already_retrieved** , **promise_already_satisfied** , **no_state** , **broken_promise** }
- enum class std::future_status { **ready** , **timeout** , **deferred** }
- enum class std::launch { **async** , **deferred** }

Functions

- **std::__basic_future< _Res >::__basic_future** (const shared_future< _Res > &) noexcept
- **std::__basic_future< _Res >::__basic_future** (future< _Res > &&) noexcept
- **std::__basic_future< _Res >::__basic_future** (shared_future< _Res > &&) noexcept
- template<typename _Fn, typename... _Args>
  future< __async_result_of< _Fn, _Args... > > std::async (_Fn &&__fn, _Args &&... __args)
- template<typename _Fn, typename... _Args>
  future< __async_result_of< _Fn, _Args... > > std::async (launch __policy, _Fn &&__fn, _Args &&... __args)
- const error_category & std::future_category () noexcept
- error_code std::make_error_code (future_errc __errc) noexcept
- error_condition std::make_error_condition (future_errc __errc) noexcept
- constexpr launch **std::operator&** (launch __x, launch __y) noexcept
- constexpr launch & **std::operator&=** (launch &__x, launch __y) noexcept
- constexpr launch **std::operator^** (launch __x, launch __y) noexcept
- constexpr launch & **std::operator^=** (launch &__x, launch __y) noexcept
- constexpr launch **std::operator|** (launch __x, launch __y) noexcept
- constexpr launch & **std::operator|=** (launch &__x, launch __y) noexcept
- constexpr launch **std::operator~** (launch __x) noexcept
- template<typename _Fun, typename _Signature = __function_guide_t<_Fun, decltype(&_Fun::operator())>>
  std::packaged_task (_Fun) -> packaged_task< _Signature >
- template<typename _Res, typename... _ArgTypes>
  std::packaged_task (_Res(*)(_ArgTypes...)) -> packaged_task< _Res(_ArgTypes...)>
- shared_future< _Res > **std::future< _Res >::share** () noexcept
- shared_future< _Res & > **std::future< _Res & >::share** () noexcept
- shared_future< void > **std::future< void >::share** () noexcept
- template<typename _Res, typename... _ArgTypes>
  void std::swap (packaged_task< _Res(_ArgTypes...)> &__x, packaged_task< _Res(_ArgTypes...)> &__y) noexcept
- template<typename _Res>
  void **std::swap** (promise< _Res > &__x, promise< _Res > &__y) noexcept

3.3.3.1 Detailed Description

Futures and promises provide support for retrieving the result from an asynchronous function, e.g. one that is running in another thread. A `std::future` represents an asynchronous result that will become ready at some later time. A consumer can wait on a future until the result is ready to be accessed.

Since

C++11

3.3.3.2 Enumeration Type Documentation

future_errc

```
enum class std::future_errc  [strong]
```

Error code for futures.

future_status

```
enum class std::future_status  [strong]
```

Status code for futures.

launch

```
enum class std::launch  [strong]
```

Launch code for futures.

3.3.3.3 Function Documentation

async() [1/2]

```
template<typename _Fn, typename...  _Args>
future< __async_result_of< _Fn, _Args...  > > std::async (
            _Fn && __fn,
            _Args &&...  __args)  [inline], [nodiscard]
```

async, potential overload

async() [2/2]

```
template<typename _Fn, typename... _Args>
future< __async_result_of< _Fn, _Args... > > std::async (
            launch __policy,
            _Fn && __fn,
            _Args &&... __args)  [nodiscard]
```

async

future_category()

```
const error_category & std::future_category ()  [noexcept]
```

Points to a statically-allocated object derived from error_category.

make_error_code()

```
error_code std::make_error_code (
            future_errc __errc)  [inline], [noexcept]
```

Overload of make_error_code for `future_errc`.

Referenced by std::filesystem::filesystem_error::what().

make_error_condition()

```
error_condition std::make_error_condition (
            future_errc __errc)  [inline], [noexcept]
```

Overload of make_error_condition for `future_errc`.

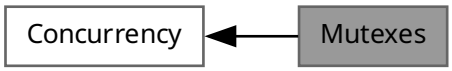
swap()

```
template<typename _Res, typename... _ArgTypes>
void std::swap (
            packaged_task< _Res(_ArgTypes...)> & __x,
            packaged_task< _Res(_ArgTypes...)> & __y)  [inline], [noexcept]
```

swap

3.3.4 Mutexes

Collaboration diagram for Mutexes:



Classes

- struct std::adopt_lock_t
- struct std::defer_lock_t
- class std::lock_guard< _Mutex >
- class std::mutex
- struct std::once_flag
- class std::recursive_mutex
- class std::recursive_timed_mutex
- class std::shared_lock< _Mutex >
- class std::shared_timed_mutex
- class std::timed_mutex
- struct std::try_to_lock_t
- class std::unique_lock< _Mutex >

Functions

- void **std::__once_proxy** (void)
- template<typename _Callable, typename... _Args>
  void std::call_once (once_flag &__once, _Callable &&__f, _Args &&... __args)
- template<typename _L1, typename _L2, typename... _L3>
  void std::lock (_L1 &__l1, _L2 &__l2, _L3 &... __l3)
- template<typename _L1, typename _L2, typename... _L3>
  int std::try_lock (_L1 &__l1, _L2 &__l2, _L3 &... __l3)

Variables

- constexpr adopt_lock_t std::adopt_lock
- constexpr defer_lock_t std::defer_lock
- constexpr try_to_lock_t std::try_to_lock

3.3.4.1 Detailed Description

Classes for mutex support.

3.3.4.2 Function Documentation

call_once()

```
template<typename _Callable, typename...  _Args>
void std::call_once (
            once_flag & __once,
            _Callable && __f,
            _Args &&...  __args)
```

Invoke a callable and synchronize with other calls using the same flag.

lock()

```
template<typename _L1, typename _L2, typename...  _L3>
void std::lock (
            _L1 & __l1,
            _L2 & __l2,
            _L3 &...  __l3)
```

Generic lock.

Parameters

_↩ _l1	Meets Lockable requirements (try_lock() may throw).
_↩ _l2	Meets Lockable requirements (try_lock() may throw).
_↩ _l3	Meets Lockable requirements (try_lock() may throw).

Exceptions

An	exception thrown by an argument's lock() or try_lock() member.

Postcondition

All arguments are locked.

All arguments are locked via a sequence of calls to lock(), try_lock() and unlock(). If this function exits via an exception any locks that were obtained will be released.

try_lock()

```
template<typename _L1, typename _L2, typename...  _L3>
int std::try_lock (
            _L1 & __l1,
            _L2 & __l2,
            _L3 &...  __l3)  [inline], [nodiscard]
```

Generic try_lock.

Parameters

_↩ _l1	Meets Lockable requirements (try_lock() may throw).
_↩ _l2	Meets Lockable requirements (try_lock() may throw).
_↩ _l3	Meets Lockable requirements (try_lock() may throw).

Returns

Returns -1 if all try_lock() calls return true. Otherwise returns a 0-based index corresponding to the argument that returned false.

Postcondition

Either all arguments are locked, or none will be.

Sequentially calls try_lock() on each argument.

3.3.4.3 Variable Documentation

adopt_lock

```
adopt_lock_t std::adopt_lock  [inline], [constexpr]
```

Tag used to make a scoped lock take ownership of a locked mutex.

defer_lock

```
defer_lock_t std::defer_lock  [inline], [constexpr]
```

Tag used to prevent a scoped lock from acquiring ownership of a mutex.

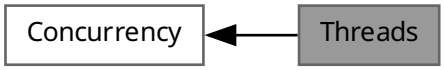
try_to_lock

```
try_to_lock_t std::try_to_lock  [inline], [constexpr]
```

Tag used to prevent a scoped lock from blocking if a mutex is locked.

3.3.5 Threads

Collaboration diagram for Threads:



Namespaces

- namespace std::this_thread

Classes

- struct std::hash< thread::id >
- class std::thread

Functions

- bool **operator==** (thread::id __x, thread::id __y) noexcept
- void **swap** (thread &__x, thread &__y) noexcept

- strong_ordering **operator**<=> (thread::id __x, thread::id __y) noexcept
- template<class _CharT, class _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__out, thread::id __id)

3.3.5.1 Detailed Description

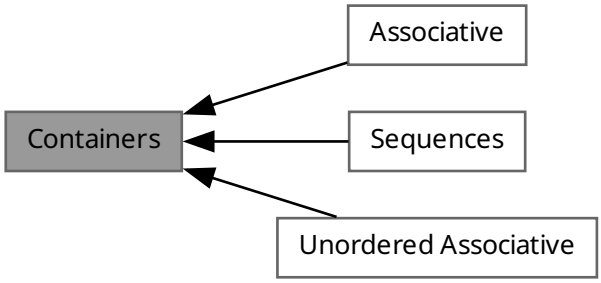
Since

C++11

Classes for thread support.

3.4 Containers

Collaboration diagram for Containers:



Topics

- Associative
- Sequences
- Unordered Associative

3.4.1 Detailed Description

Containers are collections of objects.

A container may hold any type which meets certain requirements, but the type of contained object is chosen at compile time, and all objects in a given container must be of the same type. (Polymorphism is possible by declaring a container of pointers to a base class and then populating it with pointers to instances of derived classes. Variant value types such as the `any` class from `Boost` can also be used.

All contained types must be `Assignable` and `CopyConstructible`. Specific containers may place additional requirements on the types of their contained objects.

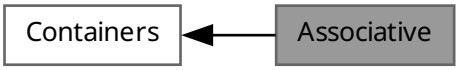
Containers manage memory allocation and deallocation themselves when storing your objects. The objects are destroyed when the container is itself destroyed. Note that if you are storing pointers in a container, `delete` is *not* automatically called on the pointers before destroying them.

All containers must meet certain requirements, summarized in `tables`.

The standard containers are further refined into Sequences and Associative Containers. Unordered Associative Containers.

3.4.2 Associative

Collaboration diagram for Associative:



Classes

- class std::map< _Key, _Tp, _Compare, _Alloc >
- class std::multimap< _Key, _Tp, _Compare, _Alloc >
- class std::multiset< _Key, _Compare, _Alloc >
- class std::set< _Key, _Compare, _Alloc >

3.4.2.1 Detailed Description

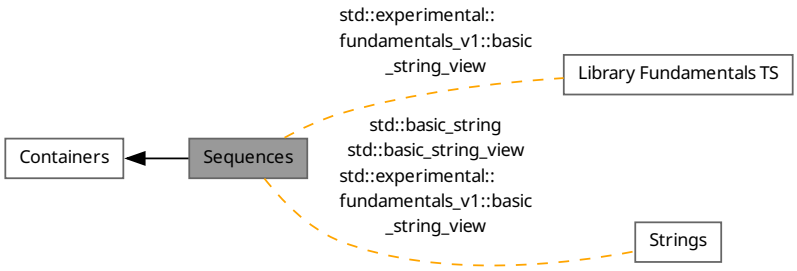
Associative containers allow fast retrieval of data based on keys.

Each container type is parameterized on a `Key` type, and an ordering relation used to sort the elements of the container.

All associative containers must meet certain requirements, summarized in `tables`.

3.4.3 Sequences

Collaboration diagram for Sequences:



Files

- file inplace_vector

Classes

- struct std::array< _Tp, _Nm >
- class std::basic_string< _CharT, _Traits, _Alloc >
- class std::basic_string_view< _CharT, _Traits >
- class std::experimental::fundamentals_v1::basic_string_view< _CharT, _Traits >
- class std::deque< _Tp, _Alloc >
- class std::forward_list< _Tp, _Alloc >
- class std::list< _Tp, _Alloc >
- class std::priority_queue< _Tp, _Sequence, _Compare >
- class std::queue< _Tp, _Sequence >
- class std::stack< _Tp, _Sequence >
- class std::vector< _Tp, _Alloc >
- class std::vector< bool, _Alloc >

3.4.3.1 Detailed Description

Sequences arrange a collection of objects into a strictly linear order.

The differences between sequences are usually due to one or both of the following:

- memory management
- algorithmic complexity

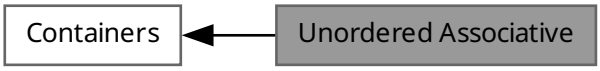
As an example of the first case, `vector` is required to use a contiguous memory layout, while other sequences such as `deque` are not.

The prime reason for choosing one sequence over another should be based on the second category of differences, algorithmic complexity. For example, if you need to perform many inserts and removals from the middle of a sequence, `list` would be ideal. But if you need to perform constant-time access to random elements of the sequence, then `list` should not be used.

All sequences must meet certain requirements, summarized in `tables`.

3.4.4 Unordered Associative

Collaboration diagram for Unordered Associative:



Classes

- class std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >
- class std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >
- class std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >
- class std::unordered_set< _Value, _Hash, _Pred, _Alloc >

3.4.4.1 Detailed Description

Unordered associative containers allow fast retrieval of data based on keys.

Each container type is parameterized on a `Key` type, a `Hash` type providing a hashing functor, and an ordering relation used to sort the elements of the container.

All unordered associative containers must meet certain requirements, summarized in `tables`.

3.5 Diagnostics

Collaboration diagram for Diagnostics:

Topics

- Exceptions

Classes

- class std::error_category
- class std::error_code
- class std::error_condition
- struct std::is_error_code_enum< _Tp >
- struct std::is_error_condition_enum< _Tp >

Functions

- const error_category & std::generic_category () noexcept
- error_code make_error_code (errc __e) noexcept
- error_condition make_error_condition (errc __e) noexcept
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const error_code &__e)
- strong_ordering operator<=> (const error_code &__lhs, const error_code &__rhs) noexcept
- strong_ordering operator<=> (const error_condition &__lhs, const error_condition &__rhs) noexcept
- bool operator== (const error_code &__lhs, const error_code &__rhs) noexcept
- bool operator== (const error_code &__lhs, const error_condition &__rhs) noexcept
- bool operator== (const error_condition &__lhs, const error_condition &__rhs) noexcept
- const error_category & std::system_category () noexcept

Variables

- template<typename _Tp>
  constexpr bool **std::is_error_code_enum_v**
- template<typename _Tp>
  constexpr bool **std::is_error_condition_enum_v**

3.5.1 Detailed Description

Components for error handling, reporting, and diagnostic operations.

3.5.2 Function Documentation

generic_category()

```
const error_category & std::generic_category ()  [noexcept]
```

Error category for `errno` error codes.

make_error_code()

```
error_code make_error_code (
            errc __e)  [related]
```

Create an `error_code` representing a standard `errc` condition.

The `std::errc` constants correspond to `errno` macros and so use the generic category.

Since

C++11

make_error_condition()

```
error_condition make_error_condition (
            errc __e)  [related]
```

Create an `error_condition` representing a standard `errc` condition.

The `std::errc` constants correspond to `errno` macros and so use the generic category.

Since

C++11

operator<<()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & operator<< (
            basic_ostream< _CharT, _Traits > & __os,
            const error_code & __e)  [related]
```

Write a std::error_code to an ostream.

Since

C++11

operator<=>() [1/2]

```
strong_ordering operator<=> (
            const error_code & __lhs,
            const error_code & __rhs)  [related]
```

Ordered comparison for std::error_code.

This defines a total order by comparing the categories, and then if they are equal comparing the values.

Since

C++11

operator<=>() [2/2]

```
strong_ordering operator<=> (
            const error_condition & __lhs,
            const error_condition & __rhs)  [related]
```

Ordered comparison for std::error_condition.

This defines a total order by comparing the categories, and then if they are equal comparing the values.

Since

C++11

operator==() [1/3]

```
bool operator== (
            const error_code & __lhs,
            const error_code & __rhs)  [related]
```

Equality comparison for std::error_code.

Returns true only if they have the same category and the same value.

Since

C++11

operator==() [2/3]

```
bool operator== (
            const error_code & __lhs,
            const error_condition & __rhs)  [related]
```

Equality comparison for std::error_code and std::error_condition.

Uses each category's `equivalent` member function to check whether the values correspond to an equivalent error in that category.

Since

C++11

operator==() [3/3]

```
bool operator== (
            const error_condition & __lhs,
            const error_condition & __rhs)  [related]
```

Equality comparison for std::error_condition.

Returns true only if they have the same category and the same value.

Since

C++11

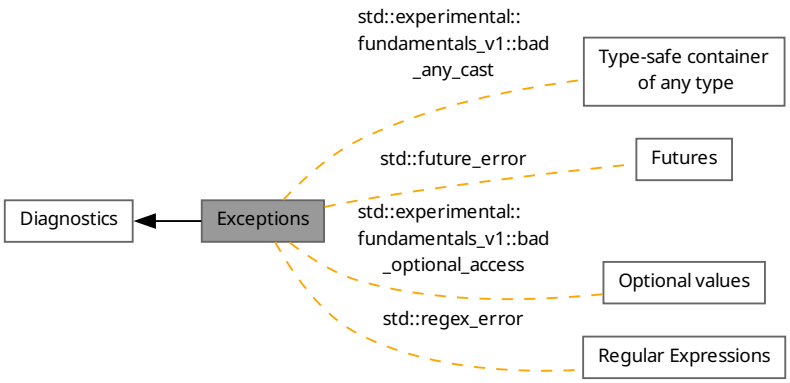
system_category()

```
const error_category & std::system_category ()  [noexcept]
```

Error category for other error codes defined by the OS.

3.5.3 Exceptions

Collaboration diagram for Exceptions:



Classes

- class __cxxabiv1::__forced_unwind
- class std::bad_alloc
- class std::experimental::fundamentals_v1::bad_any_cast
- class std::bad_cast
- class std::bad_exception
- class std::bad_function_call
- class std::experimental::fundamentals_v1::bad_optional_access
- class std::bad_typeid
- class std::bad_weak_ptr
- class std::domain_error
- class std::exception
- class std::__unspecified__::exception_ptr
- struct __gnu_cxx::forced_error
- class std::future_error
- class std::invalid_argument
- class std::length_error
- class std::logic_error
- class std::nested_exception
- class std::out_of_range
- class std::overflow_error
- class std::range_error
- class __gnu_cxx::recursive_init_error
- class std::regex_error
- class std::runtime_error
- class std::system_error
- class std::underflow_error

Typedefs

- typedef void(∗ std::terminate_handler) ()
- typedef void(∗ std::unexpected_handler) ()

Functions

- void __gnu_cxx::__verbose_terminate_handler ()
- exception_ptr std::current_exception () noexcept
- terminate_handler std::get_terminate () noexcept
- unexpected_handler std::get_unexpected () noexcept
- template<typename _Ex>
  exception_ptr std::make_exception_ptr (_Ex __ex) noexcept
- void std::rethrow_exception (exception_ptr)
- template<typename _Ex>
  void std::rethrow_if_nested (const _Ex &__ex)
- terminate_handler std::set_terminate (terminate_handler) noexcept
- unexpected_handler std::set_unexpected (unexpected_handler) noexcept
- void std::terminate () noexcept
- template<typename _Tp>
  void std::throw_with_nested (_Tp &&__t)
- bool std::uncaught_exception () noexcept
- void std::unexpected ()

Variables

- class __attribute((__abi_tag__("cxx11"))) failure typedef _Ios_Fmtflags std::ios_base::fmtflags

3.5.3.1 Detailed Description

Since

C++98

Classes and functions for reporting errors via exceptions.

3.5.3.2 Typedef Documentation

terminate_handler

```
typedef void(* std::terminate_handler) ()
```

If you write a replacement terminate handler, it must be of this type.

unexpected_handler

```
typedef void(* std::unexpected_handler) ()
```

If you write a replacement unexpected handler, it must be of this type.

3.5.3.3 Function Documentation

__verbose_terminate_handler()

```
void __gnu_cxx::__verbose_terminate_handler ()
```

A replacement for the standard terminate_handler which prints more information about the terminating exception (if any) on stderr.

Call
`std::set_terminate(__gnu_cxx::__verbose_terminate_handler)`

to use. For more info, see http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt02ch06s02.↩html

In 3.4 and later, this is on by default.

current_exception()

```
exception_ptr std::current_exception ()  [noexcept]
```

Obtain an exception_ptr to the currently handled exception.

If there is none, or the currently handled exception is foreign, return the null value.

Since

C++11

References make_exception_ptr().

Referenced by std::nested_exception::nested_exception(), and make_exception_ptr().

get_terminate()

```
terminate_handler std::get_terminate ()  [noexcept]
```

Return the current terminate handler.

get_unexpected()

```
unexpected_handler std::get_unexpected ()  [noexcept]
```

Return the current unexpected handler.

Since

C++11

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

make_exception_ptr()

```
template<typename _Ex>
exception_ptr std::make_exception_ptr (
            _Ex __ex)  [noexcept]
```

Obtain an exception_ptr pointing to a copy of the supplied object.

References current_exception().

Referenced by current_exception().

rethrow_exception()

```
void std::rethrow_exception (
            exception_ptr )
```

Throw the object pointed to by the exception_ptr.

References rethrow_exception().

Referenced by rethrow_exception(), and std::nested_exception::rethrow_nested().

rethrow_if_nested()

```
template<typename _Ex>
void std::rethrow_if_nested (
            const _Ex & __ex)  [inline]
```

Rethrow a nested exception

If `__ex` contains a `std::nested_exception` object, call its `rethrow_nested()` member to rethrow the stored exception.

After catching an exception thrown by a call to `std::throw_with_nested` this function can be used to rethrow the exception that was active when `std::throw_with_nested` was called.

Since

C++11

set_terminate()

```
terminate_handler std::set_terminate (
            terminate_handler )  [noexcept]
```

Takes a new handler function as an argument, returns the old function.

set_unexpected()

```
unexpected_handler std::set_unexpected (
            unexpected_handler )  [noexcept]
```

Takes a new handler function as an argument, returns the old function.

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

terminate()

```
void std::terminate ()  [noexcept]
```

The runtime will call this function if exception handling must be abandoned for any reason. It can also be called by the user.

Referenced by std::nested_exception::rethrow_nested().

throw_with_nested()

```
template<typename _Tp>
void std::throw_with_nested (
            _Tp && __t)  [inline]
```

Throw an exception that also stores the currently active exception.

If `_Tp` is derived from `std::nested_exception` or is not usable as a base-class, throws a copy of `__t`. Otherwise, throws an object of an implementation-defined type derived from both `_Tp` and `std::nested_exception`, containing a copy of `__t` and the result of `std::current_exception()`.

In other words, throws the argument as a new exception that contains the currently active exception nested within it. This is intended for use in a catch handler to replace the caught exception with a different type, while still preserving the original exception. When the new exception is caught, the nested exception can be rethrown by using `std::rethrow_if_nested`.

This can be used at API boundaries, for example to catch a library's internal exception type and rethrow it nested with a `std::runtime_error`, or vice versa.

Since

C++11

References forward().

uncaught_exception()

```
bool std::uncaught_exception ()  [noexcept]
```

[18.6.4]/1: 'Returns true after completing evaluation of a throw-expression until either completing initialization of the exception-declaration in the matching handler or entering `unexpected()` due to the throw; or after entering `terminate()` for any reason other than an explicit call to `terminate()`. [Note: This includes stack unwinding [15.2]. end note]'

2: 'When `uncaught_exception()` is true, throwing an exception can result in a call of 1terminate()‘ (15.5.1).’

Referenced by std::basic_ostream< _CharT, _Traits >::sentry::~sentry().

unexpected()

```
void std::unexpected ()
```

The runtime will call this function if an exception is thrown which violates the function's exception specification.

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

3.5.3.4 Variable Documentation

fmtflags

```
class __attribute ((__abi_tag__ ("cxx11"))) failure typedef _Ios_Fmtflags std::ios_base::fmtflags
```

These are thrown to indicate problems with io.

27.4.2.1.1 Class ios_base::failure

This is a bitmask type.

_Ios_Fmtflags is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type fmtflags are:

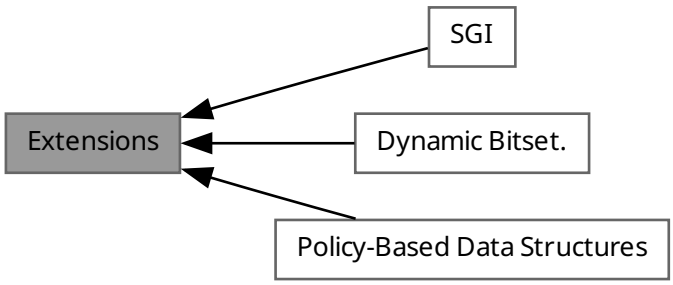
- boolalpha
- dec
- fixed
- hex
- internal
- left
- oct
- right

- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::num_put< _CharT, _OutIte
flags(), std::__detail::operator>>(), register_callback(), setf(), setf(), and unsetf().

3.6 Extensions

Collaboration diagram for Extensions:



Topics

- Dynamic Bitset.
- Policy-Based Data Structures
- SGI

Namespaces

- namespace __gnu_cxx
- namespace std::tr2

Classes

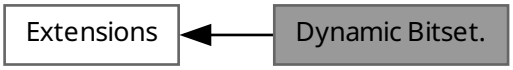
- class __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >

3.6.1 Detailed Description

Components generally useful that are not part of any standard.

3.6.2 Dynamic Bitset.

Collaboration diagram for Dynamic Bitset.:



Classes

- struct std::tr2::__dynamic_bitset_base< _WordT, _Alloc >
- class std::tr2::dynamic_bitset< _WordT, _Alloc >

Functions

- template<typename _CharT, typename _Traits, typename _Alloc1>
  void **std::tr2::dynamic_bitset< _WordT, _Alloc >::_M_copy_to_string** (std::basic_string< _CharT, _Traits, ↩
  _Alloc1 > &__str, _CharT __zero=_CharT('0'), _CharT __one=_CharT('1')) const
- template<typename _CharT, typename _Traits, typename _WordT, typename _Alloc>
  std::basic_ostream< _CharT, _Traits > & std::tr2::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const dynamic_bitset< _WordT, _Alloc > &__x)
- template<typename _CharT, typename _Traits, typename _WordT, typename _Alloc>
  std::basic_istream< _CharT, _Traits > & std::tr2::operator>> (std::basic_istream< _CharT, _Traits > &__↩
  is, dynamic_bitset< _WordT, _Alloc > &__x)

- template<typename _WordT, typename _Alloc>
  bool std::tr2::operator!= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)
- template<typename _WordT, typename _Alloc>
  bool std::tr2::operator<= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _↩
  Alloc > &__rhs)

- template<typename _WordT, typename _Alloc>
  bool std::tr2::operator> (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)
- template<typename _WordT, typename _Alloc>
  bool std::tr2::operator>= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _↩ Alloc > &__rhs)

- template<typename _WordT, typename _Alloc>
  dynamic_bitset< _WordT, _Alloc > std::tr2::operator& (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)
- template<typename _WordT, typename _Alloc>
  dynamic_bitset< _WordT, _Alloc > std::tr2::operator| (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)
- template<typename _WordT, typename _Alloc>
  dynamic_bitset< _WordT, _Alloc > std::tr2::operator^ (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)
- template<typename _WordT, typename _Alloc>
  dynamic_bitset< _WordT, _Alloc > std::tr2::operator- (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)

3.6.2.1 Detailed Description

3.6.2.2 Function Documentation

operator"!=()

```
template<typename _WordT, typename _Alloc>
bool std::tr2::operator!= (
            const dynamic_bitset< _WordT, _Alloc > & __lhs,
            const dynamic_bitset< _WordT, _Alloc > & __rhs)  [inline]
```

These comparisons for equality/inequality are, well, *bitwise*.

operator&()

```
template<typename _WordT, typename _Alloc>
dynamic_bitset< _WordT, _Alloc > std::tr2::operator& (
            const dynamic_bitset< _WordT, _Alloc > & __x,
            const dynamic_bitset< _WordT, _Alloc > & __y)  [inline]
```

Global bitwise operations on bitsets.

Parameters

_↩ _x	A bitset.
_↩ _y	A bitset of the same size as __x.

Returns

A new bitset.

These should be self-explanatory.

operator-()

```
template<typename _WordT, typename _Alloc>
dynamic_bitset< _WordT, _Alloc > std::tr2::operator- (
            const dynamic_bitset< _WordT, _Alloc > & __x,
            const dynamic_bitset< _WordT, _Alloc > & __y)  [inline]
```

Global bitwise operations on bitsets.

Parameters

_↩ _x	A bitset.
_↩ _y	A bitset of the same size as __x.

Returns

A new bitset.

These should be self-explanatory.

operator<<()

```
template<typename _CharT, typename _Traits, typename _WordT, typename _Alloc>
std::basic_ostream< _CharT, _Traits > & std::tr2::operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const dynamic_bitset< _WordT, _Alloc > & __x)  [inline]
```

Stream output operator for dynamic_bitset.

operator<=()

```
template<typename _WordT, typename _Alloc>
bool std::tr2::operator<= (
            const dynamic_bitset< _WordT, _Alloc > & __lhs,
            const dynamic_bitset< _WordT, _Alloc > & __rhs)  [inline]
```

These comparisons for equality/inequality are, well, *bitwise*.

operator>()

```
template<typename _WordT, typename _Alloc>
bool std::tr2::operator> (
            const dynamic_bitset< _WordT, _Alloc > & __lhs,
            const dynamic_bitset< _WordT, _Alloc > & __rhs)  [inline]
```

These comparisons for equality/inequality are, well, *bitwise*.

operator>=()

```
template<typename _WordT, typename _Alloc>
bool std::tr2::operator>= (
            const dynamic_bitset< _WordT, _Alloc > & __lhs,
            const dynamic_bitset< _WordT, _Alloc > & __rhs)  [inline]
```

These comparisons for equality/inequality are, well, *bitwise*.

operator>>()

```
template<typename _CharT, typename _Traits, typename _WordT, typename _Alloc>
std::basic_istream< _CharT, _Traits > & std::tr2::operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            dynamic_bitset< _WordT, _Alloc > & __x)
```

Stream input operator for dynamic_bitset.

Input will skip whitespace and only accept '0' and '1' characters. The dynamic_bitset will grow as necessary to hold the string of bits.

References std::basic_string< _CharT, _Traits, _Alloc >::empty(), std::basic_string< _CharT, _Traits, _Alloc >::push_back(), std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_string< _CharT, _Traits, _Alloc >::reserve(), std::tr2::dynamic_bitset< _WordT, _Al std::basic_ios< _CharT, _Traits >::setstate(), std::basic_string< _CharT, _Traits, _Alloc >::size(), std::tr2::dynamic_bitset< _WordT, _All and std::basic_ios< _CharT, _Traits >::widen().

operator^()

```
template<typename _WordT, typename _Alloc>
dynamic_bitset< _WordT, _Alloc > std::tr2::operator^ (
            const dynamic_bitset< _WordT, _Alloc > & __x,
            const dynamic_bitset< _WordT, _Alloc > & __y)  [inline]
```

Global bitwise operations on bitsets.

Parameters

_↩ _x	A bitset.
_↩ _y	A bitset of the same size as __x.

Returns

A new bitset.

These should be self-explanatory.

operator"|()

```
template<typename _WordT, typename _Alloc>
dynamic_bitset< _WordT, _Alloc > std::tr2::operator| (
            const dynamic_bitset< _WordT, _Alloc > & __x,
            const dynamic_bitset< _WordT, _Alloc > & __y)  [inline]
```

Global bitwise operations on bitsets.

Parameters

_↩ _x	A bitset.
_↩ _y	A bitset of the same size as __x.

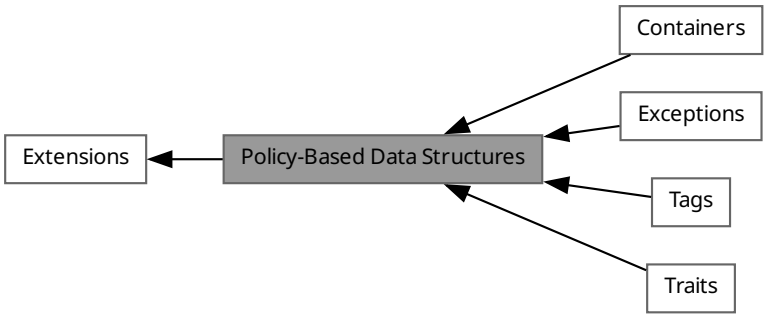
Returns

A new bitset.

These should be self-explanatory.

3.6.3 Policy-Based Data Structures

Collaboration diagram for Policy-Based Data Structures:



Topics

- Containers
- Exceptions
- Tags
- Traits

Classes

- struct __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, pairing_heap_tag, null_type >

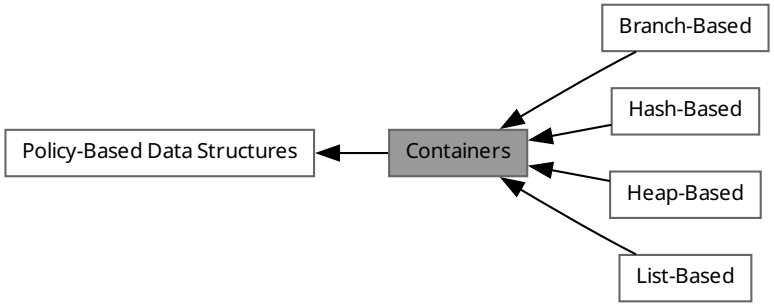
3.6.3.1 Detailed Description

This is a library of policy-based elementary data structures: associative containers and priority queues. It is designed for high-performance, flexibility, semantic safety, and conformance to the corresponding containers in std (except for some points where it differs by design).

For details, see: http://gcc.gnu.org/onlinedocs/libstdc++/ext/pb_ds/index.html

3.6.3.2 Containers

Collaboration diagram for Containers:



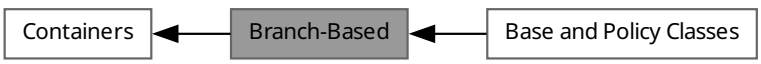
Topics

- Branch-Based
- Hash-Based
- Heap-Based
- List-Based

3.6.3.2.1 Detailed Description

3.6.3.2.2 Branch-Based

Collaboration diagram for Branch-Based:



Topics

- Base and Policy Classes

Classes

- class __gnu_pbds::basic_branch< Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc >
- class __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >
- class __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >

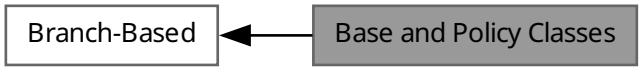
Macros

- #define **PB_DS_BRANCH_BASE**
- #define **PB_DS_TREE_BASE**
- #define **PB_DS_TREE_NODE_AND_IT_TRAITS**
- #define **PB_DS_TRIE_BASE**
- #define **PB_DS_TRIE_NODE_AND_IT_TRAITS**

Detailed Description

Base and Policy Classes

Collaboration diagram for Base and Policy Classes:



*Classes

- class __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >
- class __gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >
- class __gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >
- class __gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >

Detailed Description

3.6.3.2.3 Hash-Based

Collaboration diagram for Hash-Based:

Topics

- Base and Policy Classes

Classes

- class __gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc >
- class __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >
- class __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Al

Macros

- #define **PB_DS_CC_HASH_BASE**
- #define **PB_DS_GP_HASH_BASE**
- #define **PB_DS_HASH_BASE**

Detailed Description

Base and Policy Classes

Collaboration diagram for Base and Policy Classes:

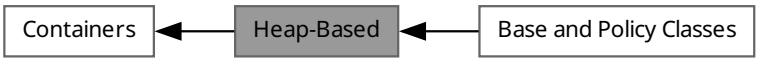
*Classes

- class __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >
- class __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_

Detailed Description

3.6.3.2.4 Heap-Based

Collaboration diagram for Heap-Based:



Topics

- Base and Policy Classes

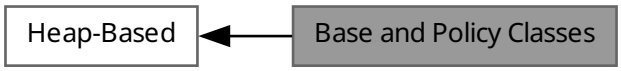
Classes

- class __gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >

Detailed Description

Base and Policy Classes

Collaboration diagram for Base and Policy Classes:



*Classes

- class __gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >
- class __gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc >
- class __gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc >
- class __gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc >
- class __gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc >

Detailed Description

3.6.3.2.5 List-Based

Collaboration diagram for List-Based:

Classes

- class __gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >

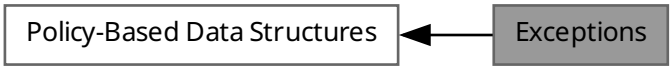
Macros

- #define **PB_DS_LU_BASE**

Detailed Description

3.6.3.3 Exceptions

Collaboration diagram for Exceptions:



Classes

- struct __gnu_pbds::container_error
- struct __gnu_pbds::insert_error
- struct __gnu_pbds::join_error
- struct __gnu_pbds::resize_error

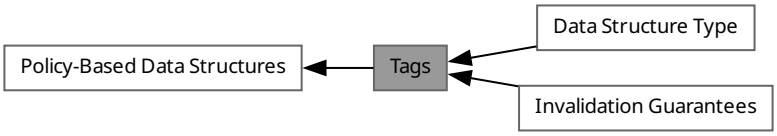
Functions

- void **__gnu_pbds::__throw_container_error** ()
- void **__gnu_pbds::__throw_insert_error** ()
- void **__gnu_pbds::__throw_join_error** ()
- void **__gnu_pbds::__throw_resize_error** ()

3.6.3.3.1 Detailed Description

3.6.3.4 Tags

Collaboration diagram for Tags:



Topics

- Data Structure Type
- Invalidation Guarantees

Classes

- struct __gnu_pbds::trivial_iterator_tag

Typedefs

- typedef void __gnu_pbds::trivial_iterator_difference_type

3.6.3.4.1 Detailed Description

3.6.3.4.2 Typedef Documentation

trivial_iterator_difference_type

```
typedef void __gnu_pbds::trivial_iterator_difference_type
```

Prohibit moving trivial iterators.

3.6.3.4.3 Data Structure Type

Collaboration diagram for Data Structure Type:

Classes

- struct __gnu_pbds::associative_tag
- struct __gnu_pbds::basic_branch_tag
- struct __gnu_pbds::basic_hash_tag
- struct __gnu_pbds::binary_heap_tag
- struct __gnu_pbds::binomial_heap_tag
- struct __gnu_pbds::cc_hash_tag
- struct __gnu_pbds::container_tag
- struct __gnu_pbds::gp_hash_tag
- struct __gnu_pbds::list_update_tag
- struct __gnu_pbds::ov_tree_tag
- struct __gnu_pbds::pairing_heap_tag
- struct __gnu_pbds::pat_trie_tag
- struct __gnu_pbds::priority_queue_tag
- struct __gnu_pbds::rb_tree_tag
- struct __gnu_pbds::rc_binomial_heap_tag
- struct __gnu_pbds::sequence_tag
- struct __gnu_pbds::splay_tree_tag
- struct __gnu_pbds::string_tag
- struct __gnu_pbds::thin_heap_tag
- struct __gnu_pbds::tree_tag
- struct __gnu_pbds::trie_tag

Detailed Description

3.6.3.4.4 Invalidation Guarantees

Collaboration diagram for Invalidation Guarantees:

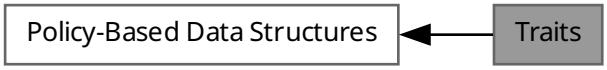
Classes

- struct __gnu_pbds::basic_invalidation_guarantee
- struct __gnu_pbds::point_invalidation_guarantee
- struct __gnu_pbds::range_invalidation_guarantee

Detailed Description

3.6.3.5 Traits

Collaboration diagram for Traits:



Classes

- struct __gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >
- struct __gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >
- struct __gnu_pbds::container_traits< Cntnr >
- struct __gnu_pbds::container_traits_base< _Tag >
- struct __gnu_pbds::container_traits_base< binary_heap_tag >
- struct __gnu_pbds::container_traits_base< binomial_heap_tag >
- struct __gnu_pbds::container_traits_base< cc_hash_tag >
- struct __gnu_pbds::container_traits_base< gp_hash_tag >
- struct __gnu_pbds::container_traits_base< list_update_tag >
- struct __gnu_pbds::container_traits_base< ov_tree_tag >
- struct __gnu_pbds::container_traits_base< pairing_heap_tag >
- struct __gnu_pbds::container_traits_base< pat_trie_tag >
- struct __gnu_pbds::container_traits_base< rb_tree_tag >
- struct __gnu_pbds::container_traits_base< rc_binomial_heap_tag >
- struct __gnu_pbds::container_traits_base< splay_tree_tag >
- struct __gnu_pbds::container_traits_base< thin_heap_tag >
- struct __gnu_pbds::detail::maybe_null_type< Key, Mapped, _Alloc, Store_Hash >
- struct __gnu_pbds::detail::maybe_null_type< Key, null_type, _Alloc, Store_Hash >
- struct __gnu_pbds::detail::no_throw_copies< Key, Mapped >
- struct __gnu_pbds::detail::no_throw_copies< Key, null_type >
- struct __gnu_pbds::null_node_update< _Tp1, _Tp2, _Tp3, _Tp4 >
- struct __gnu_pbds::null_type
- struct __gnu_pbds::detail::rebind_traits< _Alloc, T >
- struct __gnu_pbds::detail::select_value_type< Key, Mapped >

- struct __gnu_pbds::detail::select_value_type< Key, null_type >
- struct __gnu_pbds::detail::stored_data< _Tv, _Th, Store_Hash >
- struct __gnu_pbds::detail::stored_data< _Tv, _Th, false >
- struct __gnu_pbds::detail::stored_hash< _Th >
- struct __gnu_pbds::detail::stored_value< _Tv >
- struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, _BTp >
- struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, false >
- struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, true >
- struct __gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >
- struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >
- struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >
- struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >
- struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >
- struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >
- struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >
- struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, _BTp >
- struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, false >
- struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, true >
- struct __gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >
- struct __gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >
- struct __gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >
- struct __gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash >

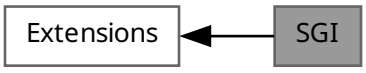
Variables

- template<typename Key, typename _Alloc, bool Store_Hash>
  null_type **__gnu_pbds::detail::maybe_null_type< Key, null_type, _Alloc, Store_Hash >::s_null_type**

3.6.3.5.1 Detailed Description

3.6.4 SGI

Collaboration diagram for SGI:



Classes

- class __gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >
- struct __gnu_cxx::constant_binary_fun< _Result, _Arg1, _Arg2 >
- struct __gnu_cxx::constant_unary_fun< _Result, _Argument >
- struct __gnu_cxx::constant_void_fun< _Result >
- class __gnu_cxx::hash_map< _Key, _Tp, _HashFn, _EqualKey, _Alloc >
- class __gnu_cxx::hash_multimap< _Key, _Tp, _HashFn, _EqualKey, _Alloc >
- class __gnu_cxx::hash_multiset< _Value, _HashFcn, _EqualKey, _Alloc >
- class __gnu_cxx::hash_set< _Value, _HashFcn, _EqualKey, _Alloc >
- struct __gnu_cxx::project1st< _Arg1, _Arg2 >
- struct __gnu_cxx::project2nd< _Arg1, _Arg2 >
- struct __gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc >
- class __gnu_cxx::rope< _CharT, _Alloc >
- struct __gnu_cxx::select1st< _Pair >
- struct __gnu_cxx::select2nd< _Pair >
- class __gnu_cxx::slist< _Tp, _Alloc >
- class __gnu_cxx::subtractive_rng
- struct __gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >
- class __gnu_cxx::unary_compose< _Operation1, _Operation2 >

Functions

- template<typename _Tp>
  const _Tp & __gnu_cxx::__median (const _Tp &__a, const _Tp &__b, const _Tp &__c)
- template<typename _Tp, typename _Compare>
  const _Tp & __gnu_cxx::__median (const _Tp &__a, const _Tp &__b, const _Tp &__c, _Compare __comp)
- constexpr size_t std::bitset< _Nb >::_Find_first () const noexcept
- constexpr size_t std::bitset< _Nb >::_Find_next (size_t __prev) const noexcept
- template<class _Operation1, class _Operation2>
  unary_compose< _Operation1, _Operation2 > __gnu_cxx::compose1 (const _Operation1 &__fn1, const _Operation2 &__fn2)
- template<class _Operation1, class _Operation2, class _Operation3>
  binary_compose< _Operation1, _Operation2, _Operation3 > __gnu_cxx::compose2 (const _Operation1 &__fn1, const _Operation2 &__fn2, const _Operation3 &__fn3)
- template<class _Result>
  constant_void_fun< _Result > __gnu_cxx::constant0 (const _Result &__val)
- template<class _Result>
  constant_unary_fun< _Result, _Result > __gnu_cxx::constant1 (const _Result &__val)
- template<class _Result>
  constant_binary_fun< _Result, _Result, _Result > __gnu_cxx::constant2 (const _Result &__val)
- template<typename _InputIterator, typename _Size, typename _OutputIterator>
  std::pair< _InputIterator, _OutputIterator > __gnu_cxx::copy_n (_InputIterator __first, _Size __count, _OutputIterator __result)
- template<typename _InputIterator, typename _Distance>
  void __gnu_cxx::distance (_InputIterator __first, _InputIterator __last, _Distance &__n)
- template<class _Tp>
  _Tp __gnu_cxx::identity_element (std::multiplies< _Tp >)
- template<class _Tp>
  _Tp __gnu_cxx::identity_element (std::plus< _Tp >)

- template<typename _InputIterator1, typename _InputIterator2>
  int __gnu_cxx::lexicographical_compare_3way (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)
- template<typename _Tp, typename _Integer>
  _Tp __gnu_cxx::power (_Tp __x, _Integer __n)
- template<typename _Tp, typename _Integer, typename _MonoidOperation>
  _Tp __gnu_cxx::power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)
- template<typename _InputIterator, typename _RandomAccessIterator>
  _RandomAccessIterator __gnu_cxx::random_sample (_InputIterator __first, _InputIterator __last, _Random↩ AccessIterator __out_first, _RandomAccessIterator __out_last)
- template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator>
  _RandomAccessIterator __gnu_cxx::random_sample (_InputIterator __first, _InputIterator __last, _Random↩ AccessIterator __out_first, _RandomAccessIterator __out_last, _RandomNumberGenerator &__rand)
- template<typename _ForwardIterator, typename _OutputIterator, typename _Distance>
  _OutputIterator __gnu_cxx::random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const _Distance __n)
- template<typename _ForwardIterator, typename _OutputIterator, typename _Distance, typename _RandomNumberGenerator>
  _OutputIterator __gnu_cxx::random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const _Distance __n, _RandomNumberGenerator &__rand)
- template<typename _InputIter, typename _Size, typename _ForwardIter>
  std::pair< _InputIter, _ForwardIter > __gnu_cxx::uninitialized_copy_n (_InputIter __first, _Size __count, _↩ ForwardIter __result)

- constexpr bitset< _Nb > & std::bitset< _Nb >::_Unchecked_set (size_t __pos) noexcept
- constexpr bitset< _Nb > & std::bitset< _Nb >::_Unchecked_set (size_t __pos, int __val) noexcept
- constexpr bitset< _Nb > & std::bitset< _Nb >::_Unchecked_reset (size_t __pos) noexcept
- constexpr bitset< _Nb > & std::bitset< _Nb >::_Unchecked_flip (size_t __pos) noexcept
- constexpr bool std::bitset< _Nb >::_Unchecked_test (size_t __pos) const noexcept

3.6.4.1 Detailed Description

Because libstdc++ based its implementation of the STL subsections of the library on the SGI 3.3 implementation, we inherited their extensions as well.

They are additionally documented in the `online documentation`, a copy of which is also shipped with the library source code (in .../docs/html/documentation.html). You can also read the documentation `on SGI's site`, which is still running even though the code is not maintained.

NB that the following notes are pulled from various comments all over the place, so they may seem stilted.

The `identity_element` functions are not part of the C++ standard; SGI provided them as an extension. Its argument is an operation, and its return value is the identity element for that operation. It is overloaded for addition and multiplication, and you can overload it for your own nefarious operations.

As an extension to the binders, SGI provided composition functors and wrapper functions to aid in their creation. The `unary_compose` functor is constructed from two functions/functors, `f` and `g`. Calling `operator()` with a single argument `x` returns `f(g(x))`. The function `compose1` takes the two functions and constructs a `unary_compose` variable for you.

`binary_compose` is constructed from three functors, `f`, `g1`, and `g2`. Its `operator()` returns `f(g1(x),g2(x))`. The function compose2 takes f, g1, and g2, and constructs the `binary_compose` instance for you. For example, if `f` returns an int, then

```
int answer = (compose2(f,g1,g2))(x);
```

is equivalent to

```
int temp1 = g1(x);
int temp2 = g2(x);
int answer = f(temp1,temp2);
```

But the first form is more compact, and can be passed around as a functor to other algorithms.
As an extension, SGI provided a functor called `identity`. When a functor is required but no operations are desired, this can be used as a pass-through. Its `operator()` returns its argument unchanged.
`select1st` and `select2nd` are extensions provided by SGI. Their `operator()s` take a `std::pair` as an argument, and return either the first member or the second member, respectively. They can be used (especially with the composition functors) to *strip* data from a sequence before performing the remainder of an algorithm.
The `operator()` of the `project1st` functor takes two arbitrary arguments and returns the first one, while `project2nd` returns the second one. They are extensions provided by SGI.
These three functors are each constructed from a single arbitrary variable/value. Later, their `operator()s` completely ignore any arguments passed, and return the stored value.

- `constant_void_fun's` `operator()` takes no arguments
- `constant_unary_fun's` `operator()` takes one argument (ignored)
- `constant_binary_fun's` `operator()` takes two arguments (ignored)

The helper creator functions `constant0`, `constant1`, and `constant2` each take a *result* argument and construct variables of the appropriate functor type.

3.6.4.2 Function Documentation

__median() [1/2]

```
template<typename _Tp>
const _Tp & __gnu_cxx::__median (
             const _Tp & __a,
             const _Tp & __b,
             const _Tp & __c)
```

Find the median of three values.

Parameters

_↩ _a	A value.
_↩ _b	A value.
_↩ _c	A value.

Returns

One of `a`, `b` or `c`.

If `{l`,m,n} is some convolution of `{a`,b,c} such that `l<=m<=n` then the value returned will be `m`. This is an SGI extension.

__median() [2/2]

```
template<typename _Tp, typename _Compare>
const _Tp & __gnu_cxx::__median (
             const _Tp & __a,
             const _Tp & __b,
             const _Tp & __c,
             _Compare __comp)
```

Find the median of three values using a predicate for comparison.

Parameters

__a	A value.
__b	A value.
__c	A value.
__comp	A binary predicate.

Returns

One of `a`, `b` or `c`.

If `{l,m,n}` is some convolution of `{a,b,c}` such that `comp(l,m)` and `comp(m,n)` are both true then the value returned will be `m`. This is an SGI extension.

_Find_first()

```
template<size_t _Nb>
size_t std::bitset< _Nb >::_Find_first () const  [inline], [constexpr], [noexcept]
```

Finds the index of the first "on" bit.

Returns

The index of the first bit set, or size() if not found.

See also

_Find_next

_Find_next()

```
template<size_t _Nb>
size_t std::bitset< _Nb >::_Find_next (
            size_t __prev) const  [inline], [constexpr], [noexcept]
```

Finds the index of the next "on" bit after prev.

Returns

The index of the next bit set, or size() if not found.

Parameters

__prev	Where to start searching.

See also

_Find_first

_Unchecked_flip()

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::_Unchecked_flip (
            size_t __pos)  [inline], [constexpr], [noexcept]
```

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

_Unchecked_reset()

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::_Unchecked_reset (
            size_t __pos)  [inline], [constexpr], [noexcept]
```

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

_Unchecked_set() [1/2]

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::_Unchecked_set (
            size_t __pos)  [inline], [constexpr], [noexcept]
```

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

_Unchecked_set() [2/2]

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::_Unchecked_set (
            size_t __pos,
            int __val)  [inline], [constexpr], [noexcept]
```

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

_Unchecked_test()

```
template<size_t _Nb>
bool std::bitset< _Nb >::_Unchecked_test (
            size_t __pos) const  [inline], [constexpr], [noexcept]
```

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

compose1()

```
template<class _Operation1, class _Operation2>
unary_compose< _Operation1, _Operation2 > __gnu_cxx::compose1 (
            const _Operation1 & __fn1,
            const _Operation2 & __fn2)  [inline]
```

An SGI extension .

compose2()

```
template<class _Operation1, class _Operation2, class _Operation3>
binary_compose< _Operation1, _Operation2, _Operation3 > __gnu_cxx::compose2 (
            const _Operation1 & __fn1,
            const _Operation2 & __fn2,
            const _Operation3 & __fn3)  [inline]
```

An SGI extension .

constant0()

```
template<class _Result>
constant_void_fun< _Result > __gnu_cxx::constant0 (
            const _Result & __val)  [inline]
```

An SGI extension .

constant1()

```
template<class _Result>
constant_unary_fun< _Result, _Result > __gnu_cxx::constant1 (
            const _Result & __val)  [inline]
```

An SGI extension .

constant2()

```
template<class _Result>
constant_binary_fun< _Result, _Result, _Result > __gnu_cxx::constant2 (
            const _Result & __val)  [inline]
```

An SGI extension .

copy_n()

```
template<typename _InputIterator, typename _Size, typename _OutputIterator>
std::pair< _InputIterator, _OutputIterator > __gnu_cxx::copy_n (
            _InputIterator __first,
            _Size __count,
            _OutputIterator __result)  [inline]
```

Copies the range [first,first+count) into [result,result+count).

Parameters

__first	An input iterator.
__count	The number of elements to copy.
__result	An output iterator.

Returns

A std::pair composed of first+count and result+count.

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

distance()

```
template<typename _InputIterator, typename _Distance>
void __gnu_cxx::distance (
            _InputIterator __first,
            _InputIterator __last,
            _Distance & __n)  [inline]
```

This is an SGI extension.

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

identity_element() [1/2]

```
template<class _Tp>
_Tp __gnu_cxx::identity_element (
            std::multiplies< _Tp > )  [inline]
```

An SGI extension .

identity_element() [2/2]

```
template<class _Tp>
_Tp __gnu_cxx::identity_element (
            std::plus< _Tp > )  [inline]
```

An SGI extension .

lexicographical_compare_3way()

```
template<typename _InputIterator1, typename _InputIterator2>
int __gnu_cxx::lexicographical_compare_3way (
            _InputIterator1 __first1,
            _InputIterator1 __last1,
            _InputIterator2 __first2,
            _InputIterator2 __last2)
```

memcmp on steroids.

Parameters

__first1	An input iterator.
__last1	An input iterator.
__first2	An input iterator.
__last2	An input iterator.

Returns

An int, as with memcmp.

The return value will be less than zero if the first range is *lexigraphically less than* the second, greater than zero if the second range is *lexigraphically less than* the first, and zero otherwise. This is an SGI extension.

power() [1/2]

```
template<typename _Tp, typename _Integer>
_Tp __gnu_cxx::power (
            _Tp __x,
            _Integer __n)  [inline]
```

This is an SGI extension.

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

power() [2/2]

```
template<typename _Tp, typename _Integer, typename _MonoidOperation>
_Tp __gnu_cxx::power (
            _Tp __x,
            _Integer __n,
            _MonoidOperation __monoid_op)  [inline]
```

This is an SGI extension.

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

random_sample() [1/2]

```
template<typename _InputIterator, typename _RandomAccessIterator>
_RandomAccessIterator __gnu_cxx::random_sample (
            _InputIterator __first,
            _InputIterator __last,
            _RandomAccessIterator __out_first,
            _RandomAccessIterator __out_last)  [inline]
```

This is an SGI extension.

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

random_sample() [2/2]

```
template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator>
_RandomAccessIterator __gnu_cxx::random_sample (
            _InputIterator __first,
            _InputIterator __last,
            _RandomAccessIterator __out_first,
            _RandomAccessIterator __out_last,
            _RandomNumberGenerator & __rand)  [inline]
```

This is an SGI extension.

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

random_sample_n() [1/2]

```
template<typename _ForwardIterator, typename _OutputIterator, typename _Distance>
_OutputIterator __gnu_cxx::random_sample_n (
            _ForwardIterator __first,
            _ForwardIterator __last,
            _OutputIterator __out,
            const _Distance __n)
```

This is an SGI extension.

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

random_sample_n() [2/2]

```
template<typename _ForwardIterator, typename _OutputIterator, typename _Distance, typename _↩
RandomNumberGenerator>
_OutputIterator __gnu_cxx::random_sample_n (
            _ForwardIterator __first,
            _ForwardIterator __last,
            _OutputIterator __out,
            const _Distance __n,
            _RandomNumberGenerator & __rand)
```

This is an SGI extension.

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

uninitialized_copy_n()

```
template<typename _InputIter, typename _Size, typename _ForwardIter>
std::pair< _InputIter, _ForwardIter > __gnu_cxx::uninitialized_copy_n (
            _InputIter __first,
            _Size __count,
            _ForwardIter __result)  [inline]
```

Copies the range [first,last) into result.

Parameters

__first	An input iterator.
__count	Length
__result	An output iterator.

Returns

__result + (__first + __count)

Like copy(), but does not require an initialized output range.

3.7 Filesystem

Files

- file filesystem

Classes

- class std::filesystem::directory_entry
- class std::filesystem::directory_iterator
- class std::filesystem::file_status
- class std::filesystem::filesystem_error
- class std::filesystem::path::iterator
- class std::filesystem::path
- class std::filesystem::recursive_directory_iterator
- struct std::filesystem::space_info

Typedefs

- using std::filesystem::file_time_type

Enumerations

- enum class std::filesystem::copy_options : unsigned short {
  none , **skip_existing** , **overwrite_existing** , **update_existing** ,
  recursive , **copy_symlinks** , **skip_symlinks** , **directories_only** ,
  create_symlinks , **create_hard_links** }
- enum class std::filesystem::directory_options : unsigned char { **none** , **follow_directory_symlink** , **skip_permission_denied** }
- enum class std::filesystem::file_type : signed char {
  none , **not_found** , **regular** , **directory** ,
  symlink , **block** , **character** , **fifo** ,
  socket , **unknown** }

- enum class std::filesystem::perm_options : unsigned { **replace** , **add** , **remove** , **nofollow** }
- enum class std::filesystem::perms : unsigned {
  none , **owner_read** , **owner_write** , **owner_exec** ,
  owner_all , **group_read** , **group_write** , **group_exec** ,
  group_all , **others_read** , **others_write** , **others_exec** ,
  others_all , **all** , **set_uid** , **set_gid** ,
  sticky_bit , **mask** , **unknown** }

Functions

- path **std::filesystem::absolute** (const path &__p)
- path **std::filesystem::absolute** (const path &__p, error_code &__ec)
- path & **std::filesystem::path::assign** (string_type &&__source)
- iterator **std::filesystem::path::begin** () const noexcept
- path **std::filesystem::canonical** (const path &__p)
- path **std::filesystem::canonical** (const path &__p, error_code &__ec)
- int **std::filesystem::path::compare** (const string_type &__s) const noexcept
- int **std::filesystem::path::compare** (const value_type *__s) const noexcept
- void **std::filesystem::copy** (const path &__from, const path &__to)
- void **std::filesystem::copy** (const path &__from, const path &__to, copy_options __options)
- void **std::filesystem::copy** (const path &__from, const path &__to, copy_options __options, error_code &)
- void **std::filesystem::copy** (const path &__from, const path &__to, error_code &__ec)
- bool **std::filesystem::copy_file** (const path &__from, const path &__to)
- bool **std::filesystem::copy_file** (const path &__from, const path &__to, copy_options __option)
- bool **std::filesystem::copy_file** (const path &__from, const path &__to, copy_options __option, error_code &)
- bool **std::filesystem::copy_file** (const path &__from, const path &__to, error_code &__ec)
- void **std::filesystem::copy_symlink** (const path &__existing_symlink, const path &__new_symlink)
- void **std::filesystem::copy_symlink** (const path &__existing_symlink, const path &__new_symlink, error_code &__ec) noexcept
- bool **std::filesystem::create_directories** (const path &__p)
- bool **std::filesystem::create_directories** (const path &__p, error_code &__ec)
- bool **std::filesystem::create_directory** (const path &__p)
- bool **std::filesystem::create_directory** (const path &__p, const path &__attributes)
- bool **std::filesystem::create_directory** (const path &__p, const path &__attributes, error_code &__ec) noexcept
- bool **std::filesystem::create_directory** (const path &__p, error_code &__ec) noexcept
- void **std::filesystem::create_directory_symlink** (const path &__to, const path &__new_symlink)
- void **std::filesystem::create_directory_symlink** (const path &__to, const path &__new_symlink, error_code &__ec) noexcept
- void **std::filesystem::create_hard_link** (const path &__to, const path &__new_hard_link)
- void **std::filesystem::create_hard_link** (const path &__to, const path &__new_hard_link, error_code &__ec) noexcept
- void **std::filesystem::create_symlink** (const path &__to, const path &__new_symlink)
- void **std::filesystem::create_symlink** (const path &__to, const path &__new_symlink, error_code &__ec) noexcept
- path **std::filesystem::current_path** ()
- void **std::filesystem::current_path** (const path &__p)
- void **std::filesystem::current_path** (const path &__p, error_code &__ec) noexcept
- path **std::filesystem::current_path** (error_code &__ec)
- iterator **std::filesystem::path::end** () const noexcept
- bool **std::filesystem::equivalent** (const path &__p1, const path &__p2)
- bool **std::filesystem::equivalent** (const path &__p1, const path &__p2, error_code &__ec) noexcept
- bool **std::filesystem::exists** (const path &__p)

- bool **std::filesystem::exists** (const path &__p, error_code &__ec) noexcept
- bool **std::filesystem::exists** (file_status) noexcept
- path **std::filesystem::path::extension** () const
- uintmax_t **std::filesystem::file_size** (const path &)
- uintmax_t **std::filesystem::file_size** (const path &, error_code &) noexcept
- path **std::filesystem::path::filename** () const
- std::string **std::filesystem::path::generic_string** () const
- template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>> std::basic_string< _CharT, _Traits, _Allocator > **std::filesystem::path::generic_string** (const _Allocator &__a=_Allocator()) const
- std::u16string **std::filesystem::path::generic_u16string** () const
- std::u32string **std::filesystem::path::generic_u32string** () const
- std::string **std::filesystem::path::generic_u8string** () const
- std::wstring **std::filesystem::path::generic_wstring** () const
- uintmax_t **std::filesystem::hard_link_count** (const path &)
- uintmax_t **std::filesystem::hard_link_count** (const path &, error_code &) noexcept
- bool **std::filesystem::path::has_extension** () const noexcept
- bool **std::filesystem::path::has_stem** () const noexcept
- size_t **std::filesystem::hash_value** (const path &__p) noexcept
- bool **std::filesystem::path::is_absolute** () const noexcept
- bool **std::filesystem::is_block_file** (const path &__p)
- bool **std::filesystem::is_block_file** (const path &__p, error_code &__ec) noexcept
- bool **std::filesystem::is_block_file** (file_status __s) noexcept
- bool **std::filesystem::is_character_file** (const path &__p)
- bool **std::filesystem::is_character_file** (const path &__p, error_code &__ec) noexcept
- bool **std::filesystem::is_character_file** (file_status __s) noexcept
- bool **std::filesystem::is_directory** (const path &__p)
- bool **std::filesystem::is_directory** (const path &__p, error_code &__ec) noexcept
- bool **std::filesystem::is_directory** (file_status __s) noexcept
- bool **std::filesystem::is_empty** (const path &__p)
- bool **std::filesystem::is_empty** (const path &__p, error_code &__ec)
- bool **std::filesystem::is_fifo** (const path &__p)
- bool **std::filesystem::is_fifo** (const path &__p, error_code &__ec) noexcept
- bool **std::filesystem::is_fifo** (file_status __s) noexcept
- bool **std::filesystem::is_other** (const path &__p)
- bool **std::filesystem::is_other** (const path &__p, error_code &__ec) noexcept
- bool **std::filesystem::is_other** (file_status) noexcept
- bool **std::filesystem::is_regular_file** (const path &__p)
- bool **std::filesystem::is_regular_file** (const path &__p, error_code &__ec) noexcept
- bool **std::filesystem::is_regular_file** (file_status) noexcept
- bool **std::filesystem::is_socket** (const path &__p)
- bool **std::filesystem::is_socket** (const path &__p, error_code &__ec) noexcept
- bool **std::filesystem::is_socket** (file_status __s) noexcept
- bool **std::filesystem::is_symlink** (const path &__p)
- bool **std::filesystem::is_symlink** (const path &__p, error_code &__ec) noexcept
- bool **std::filesystem::is_symlink** (file_status) noexcept
- file_time_type **std::filesystem::last_write_time** (const path &)
- file_time_type **std::filesystem::last_write_time** (const path &, error_code &) noexcept
- void **std::filesystem::last_write_time** (const path &__p, file_time_type __new_time)
- void **std::filesystem::last_write_time** (const path &__p, file_time_type __new_time, error_code &__ec) noexcept

- path & **std::filesystem::path::make_preferred** ()
- copy_options & **std::filesystem::operator&=** (copy_options &__x, copy_options __y) noexcept
- reference **std::filesystem::path::iterator::operator*** () const noexcept
- iterator & **std::filesystem::path::iterator::operator++** () noexcept
- template<typename _CharT>
  __detail::_Path2< _CharT * > & **std::filesystem::path::operator+=** (_CharT __x)
- path & **std::filesystem::path::operator+=** (basic_string_view< value_type > __x)
- path & **std::filesystem::path::operator+=** (const string_type &__x)
- path & **std::filesystem::path::operator+=** (const value_type *__x)
- path & **std::filesystem::path::operator+=** (value_type __x)
- iterator & **std::filesystem::path::iterator::operator--** () noexcept
- path & **std::filesystem::path::operator=** (path &&) noexcept
- path & **std::filesystem::path::operator=** (string_type &&__source)
- constexpr copy_options **std::filesystem::operator^** (copy_options __x, copy_options __y) noexcept
- copy_options & **std::filesystem::operator^=** (copy_options &__x, copy_options __y) noexcept
- constexpr copy_options **std::filesystem::operator|** (copy_options __x, copy_options __y) noexcept
- copy_options & **std::filesystem::operator|=** (copy_options &__x, copy_options __y) noexcept
- constexpr copy_options **std::filesystem::operator~** (copy_options __x) noexcept
- void **std::filesystem::permissions** (const path &, perms, perm_options, error_code &) noexcept
- void **std::filesystem::permissions** (const path &__p, perms __prms, error_code &__ec) noexcept
- void **std::filesystem::permissions** (const path &__p, perms __prms, perm_options __opts=perm_options::replace)
- path **std::filesystem::proximate** (const path &__p, const path &__base, error_code &__ec)
- path **std::filesystem::proximate** (const path &__p, const path &__base=current_path())
- path **std::filesystem::proximate** (const path &__p, error_code &__ec)
- path **std::filesystem::read_symlink** (const path &__p)
- path **std::filesystem::read_symlink** (const path &__p, error_code &__ec)
- path **std::filesystem::relative** (const path &__p, const path &__base, error_code &__ec)
- path **std::filesystem::relative** (const path &__p, const path &__base=current_path())
- path **std::filesystem::relative** (const path &__p, error_code &__ec)
- bool **std::filesystem::remove** (const path &, error_code &) noexcept
- bool **std::filesystem::remove** (const path &__p)
- uintmax_t **std::filesystem::remove_all** (const path &)
- uintmax_t **std::filesystem::remove_all** (const path &, error_code &)
- void **std::filesystem::rename** (const path &__from, const path &__to)
- void **std::filesystem::rename** (const path &__from, const path &__to, error_code &__ec) noexcept
- void **std::filesystem::resize_file** (const path &__p, uintmax_t __size)
- void **std::filesystem::resize_file** (const path &__p, uintmax_t __size, error_code &__ec) noexcept
- space_info **std::filesystem::space** (const path &__p)
- space_info **std::filesystem::space** (const path &__p, error_code &__ec) noexcept
- file_status **std::filesystem::status** (const path &)
- file_status **std::filesystem::status** (const path &, error_code &) noexcept
- bool **std::filesystem::status_known** (file_status) noexcept
- path **std::filesystem::path::stem** () const
- std::string **std::filesystem::path::string** () const
- template<typename _CharT, typename _Traits, typename _Allocator>
  basic_string< _CharT, _Traits, _Allocator > **std::filesystem::path::string** (const _Allocator &__a) const
- void **swap** (path &__lhs, path &__rhs) noexcept
- void **std::filesystem::path::swap** (path &__rhs) noexcept
- file_status **std::filesystem::symlink_status** (const path &)
- file_status **std::filesystem::symlink_status** (const path &, error_code &) noexcept

- path **std::filesystem::temp_directory_path** ()
- path **std::filesystem::temp_directory_path** (error_code &__ec)
- std::u16string **std::filesystem::path::u16string** () const
- std::u32string **std::filesystem::path::u32string** () const
- template<typename _InputIterator, typename _Require = __detail::_Path2<_InputIterator>, typename _CharT = __detail::__value_type_is_char_or_char8_t<_InputIterator>>
  path u8path (_InputIterator __first, _InputIterator __last)
- template<typename _Source, typename _Require = __detail::_Path<_Source>, typename _CharT = __detail::__value_type_is_char_or_char8_t<_Source>>
  path u8path (const _Source &__source)
- std::string **std::filesystem::path::u8string** () const
- path **std::filesystem::weakly_canonical** (const path &__p)
- path **std::filesystem::weakly_canonical** (const path &__p, error_code &__ec)
- std::wstring **std::filesystem::path::wstring** () const

- constexpr perms **std::filesystem::operator|** (perms __x, perms __y) noexcept
- constexpr perms **std::filesystem::operator^** (perms __x, perms __y) noexcept
- constexpr perms **std::filesystem::operator~** (perms __x) noexcept
- perms & **std::filesystem::operator&=** (perms &__x, perms __y) noexcept
- perms & **std::filesystem::operator|=** (perms &__x, perms __y) noexcept
- perms & **std::filesystem::operator^=** (perms &__x, perms __y) noexcept

- constexpr perm_options **std::filesystem::operator|** (perm_options __x, perm_options __y) noexcept
- constexpr perm_options **std::filesystem::operator^** (perm_options __x, perm_options __y) noexcept
- constexpr perm_options **std::filesystem::operator~** (perm_options __x) noexcept
- perm_options & **std::filesystem::operator&=** (perm_options &__x, perm_options __y) noexcept
- perm_options & **std::filesystem::operator|=** (perm_options &__x, perm_options __y) noexcept
- perm_options & **std::filesystem::operator^=** (perm_options &__x, perm_options __y) noexcept

- constexpr directory_options **std::filesystem::operator|** (directory_options __x, directory_options __y) noexcept
- constexpr directory_options **std::filesystem::operator^** (directory_options __x, directory_options __y) noexcept
- constexpr directory_options **std::filesystem::operator~** (directory_options __x) noexcept
- directory_options & **std::filesystem::operator&=** (directory_options &__x, directory_options __y) noexcept
- directory_options & **std::filesystem::operator|=** (directory_options &__x, directory_options __y) noexcept
- directory_options & **std::filesystem::operator^=** (directory_options &__x, directory_options __y) noexcept

- directory_iterator begin (directory_iterator __iter) noexcept
- directory_iterator end (directory_iterator) noexcept

- recursive_directory_iterator begin (recursive_directory_iterator __iter) noexcept
- recursive_directory_iterator end (recursive_directory_iterator) noexcept

3.7.1 Detailed Description

3.7.2 Typedef Documentation

file_time_type

```
using std::filesystem::file_time_type
```

The type used for file timestamps.

3.7.3 Enumeration Type Documentation

copy_options

```
enum class std::filesystem::copy_options :  unsigned short  [strong]
```

Bitmask type controlling effects of `filesystem::copy`

directory_options

enum class std::filesystem::directory_options : unsigned char [strong]
Bitmask type controlling directory iteration.

file_type

enum class std::filesystem::file_type : signed char [strong]
Enumerated type representing the type of a file.

perm_options

enum class std::filesystem::perm_options : unsigned [strong]
Bitmask type controlling changes to permissions.

perms

enum class std::filesystem::perms : unsigned [strong]
Bitmask type representing file access permissions.

3.7.4 Function Documentation

begin() [1/2]

directory_iterator begin (
directory_iterator __*iter*) [related]
Enable range-based `for` using directory_iterator.

Since

C++17

e.g. `for (auto& entry : std::filesystem::directory_iterator(".")) ...`

begin() [2/2]

recursive_directory_iterator begin (
recursive_directory_iterator __*iter*) [related]
Enable range-based `for` using recursive_directory_iterator.

Since

C++17

e.g. `for (auto& entry : recursive_directory_iterator(".")) ...`

end() [1/2]

directory_iterator end (
directory_iterator) [related]
Return a past-the-end directory_iterator.

end() [2/2]

recursive_directory_iterator end (
recursive_directory_iterator) [related]
Return a past-the-end recursive_directory_iterator.

u8path() [1/2]

```
template<typename _InputIterator, typename _Require = __detail::_Path2<_InputIterator>, typename
_CharT = __detail::__value_type_is_char_or_char8_t<_InputIterator>>
path u8path (
            _InputIterator __first,
            _InputIterator __last)  [related]
```

Create a path from a UTF-8-encoded sequence of char

Since

C++17

References u8path().
Referenced by u8path(), and u8path().

u8path() [2/2]

```
template<typename _Source, typename _Require = __detail::_Path<_Source>, typename _CharT = __↩
detail::__value_type_is_char_or_char8_t<_Source>>
path u8path (
            const _Source & __source)  [related]
```

Create a path from a UTF-8-encoded sequence of char

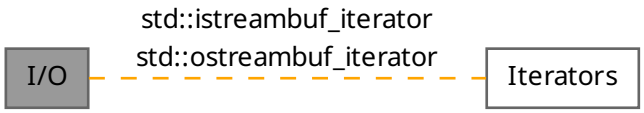
Since

C++17

References u8path().

3.8 I/O

Collaboration diagram for I/O:



Classes

- class std::basic_filebuf< _CharT, _Traits >
- class std::basic_fstream< _CharT, _Traits >
- class std::basic_ifstream< _CharT, _Traits >
- class std::basic_ios< _CharT, _Traits >
- class std::basic_iostream< _CharT, _Traits >
- class std::basic_istream< _CharT, _Traits >
- class std::basic_istringstream< _CharT, _Traits, _Alloc >
- class std::basic_ofstream< _CharT, _Traits >
- class std::basic_ostream< _CharT, _Traits >

- class std::basic_ostringstream< _CharT, _Traits, _Alloc >
- class std::basic_streambuf< _CharT, _Traits >
- class std::basic_stringbuf< _CharT, _Traits, _Alloc >
- class std::basic_stringstream< _CharT, _Traits, _Alloc >
- class std::ios_base
- class std::istreambuf_iterator< _CharT, _Traits >
- class std::ostreambuf_iterator< _CharT, _Traits >
- class __gnu_cxx::stdio_filebuf< _CharT, _Traits >
- class __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >

Typedefs

- typedef basic_filebuf< char > std::filebuf
- typedef basic_fstream< char > std::fstream
- typedef basic_ifstream< char > std::ifstream
- typedef basic_ios< char > std::ios
- typedef basic_iostream< char > std::iostream
- typedef basic_istream< char > std::istream
- typedef basic_istringstream< char > std::istringstream
- typedef basic_ofstream< char > std::ofstream
- typedef basic_ostream< char > std::ostream
- typedef basic_ostringstream< char > std::ostringstream
- typedef basic_streambuf< char > std::streambuf
- typedef basic_stringbuf< char > std::stringbuf
- typedef basic_stringstream< char > std::stringstream
- typedef basic_filebuf< wchar_t > std::wfilebuf
- typedef basic_fstream< wchar_t > std::wfstream
- typedef basic_ifstream< wchar_t > std::wifstream
- typedef basic_ios< wchar_t > std::wios
- typedef basic_iostream< wchar_t > std::wiostream
- typedef basic_istream< wchar_t > std::wistream
- typedef basic_istringstream< wchar_t > std::wistringstream
- typedef basic_ofstream< wchar_t > std::wofstream
- typedef basic_ostream< wchar_t > std::wostream
- typedef basic_ostringstream< wchar_t > std::wostringstream
- typedef basic_streambuf< wchar_t > std::wstreambuf
- typedef basic_stringbuf< wchar_t > std::wstringbuf
- typedef basic_stringstream< wchar_t > std::wstringstream

3.8.1 Detailed Description

Nearly all of the I/O classes are parameterized on the type of characters they read and write. (The major exception is ios_base at the top of the hierarchy.) This is a change from pre-Standard streams, which were not templates.
For ease of use and compatibility, all of the basic_* I/O-related classes are given typedef names for both of the builtin character widths (wide and narrow). The typedefs are the same as the pre-Standard names, for example:

```
typedef basic_ifstream<char>  ifstream;
```

Because properly forward-declaring these classes can be difficult, you should not do it yourself. Instead, include the <iosfwd> header, which contains only declarations of all the I/O classes as well as the typedefs. Trying to forward-declare the typedefs themselves (e.g., `class ostream;`) is not valid ISO C++.
For more specific declarations, see https://gcc.gnu.org/onlinedocs/libstdc++/manual/io.html#std.io.objects

3.8.2 Typedef Documentation

filebuf

typedef basic_filebuf<char> std::filebuf
Class for char file buffers.

fstream

typedef basic_fstream<char> std::fstream
Class for char mixed input and output file streams.

ifstream

typedef basic_ifstream<char> std::ifstream
Class for char input file streams.

ios

typedef basic_ios<char> std::ios
Base class for char streams.

iostream

typedef basic_iostream<char> std::iostream
Base class for char mixed input and output streams.

istream

typedef basic_istream<char> std::istream
Base class for char input streams.

istringstream

typedef basic_istringstream<char> std::istringstream
Class for char input memory streams.

ofstream

typedef basic_ofstream<char> std::ofstream
Class for char output file streams.

ostream

typedef basic_ostream<char> std::ostream
Base class for char output streams.

ostringstream

typedef basic_ostringstream<char> std::ostringstream
Class for char output memory streams.

streambuf

typedef basic_streambuf<char> std::streambuf
Base class for char buffers.

stringbuf

`typedef basic_stringbuf<char> std::stringbuf`
Class for `char` memory buffers.

stringstream

`typedef basic_stringstream<char> std::stringstream`
Class for `char` mixed input and output memory streams.

wfilebuf

`typedef basic_filebuf<wchar_t> std::wfilebuf`
Class for `wchar_t` file buffers.

wfstream

`typedef basic_fstream<wchar_t> std::wfstream`
Class for `wchar_t` mixed input and output file streams.

wifstream

`typedef basic_ifstream<wchar_t> std::wifstream`
Class for `wchar_t` input file streams.

wios

`typedef basic_ios<wchar_t> std::wios`
Base class for `wchar_t` streams.

wiostream

`typedef basic_iostream<wchar_t> std::wiostream`
Base class for `wchar_t` mixed input and output streams.

wistream

`typedef basic_istream<wchar_t> std::wistream`
Base class for `wchar_t` input streams.

wistringstream

`typedef basic_istringstream<wchar_t> std::wistringstream`
Class for `wchar_t` input memory streams.

wofstream

`typedef basic_ofstream<wchar_t> std::wofstream`
Class for `wchar_t` output file streams.

wostream

`typedef basic_ostream<wchar_t> std::wostream`
Base class for `wchar_t` output streams.

wostringstream

typedef basic_ostringstream<wchar_t> std::wostringstream
Class for wchar_t output memory streams.

wstreambuf

typedef basic_streambuf<wchar_t> std::wstreambuf
Base class for wchar_t buffers.

wstringbuf

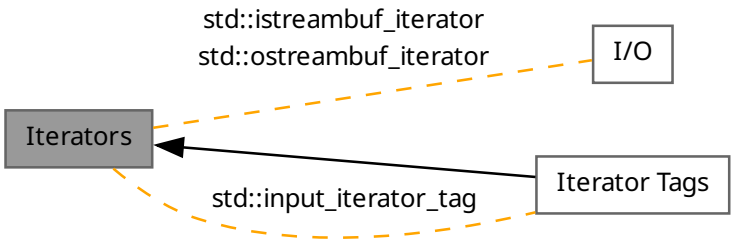
typedef basic_stringbuf<wchar_t> std::wstringbuf
Class for wchar_t memory buffers.

wstringstream

typedef basic_stringstream<wchar_t> std::wstringstream
Class for wchar_t mixed input and output memory streams.

3.9 Iterators

Collaboration diagram for Iterators:



Topics

- Iterator Tags

Classes

- class std::back_insert_iterator< _Container >
- class std::common_iterator< _It, _Sent >
- class std::counted_iterator< _It >
- class std::front_insert_iterator< _Container >
- struct std::input_iterator_tag
- class std::insert_iterator< _Container >
- class std::istream_iterator< _Tp, _CharT, _Traits, _Dist >
- class std::istreambuf_iterator< _CharT, _Traits >
- struct std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >

- struct std::iterator_traits< _Iterator >
- struct std::iterator_traits< _Tp * >
- class std::move_iterator< _Iterator >
- class std::move_sentinel< _Sent >
- class std::ostream_iterator< _Tp, _CharT, _Traits >
- class std::ostreambuf_iterator< _CharT, _Traits >
- class std::reverse_iterator< _Iterator >

Macros

- #define **_GLIBCXX_MAKE_MOVE_IF_NOEXCEPT_ITERATOR**(_Iter)
- #define **_GLIBCXX_MAKE_MOVE_ITERATOR**(_Iter)

Functions

- template<bool _IsMove, typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type **std::__copy_move_a2** (_CharT *__first, _CharT *__last, ostreambuf_iterator< _CharT > __result)
- template<bool _IsMove, typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type **std::__copy_move_a2** (const _CharT *__first, const _CharT *__last, ostreambuf_iterator< _CharT > __result)
- template<bool _IsMove, typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type **std::__copy_move_a2** (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, _CharT *__result)
- template<typename _CharT, typename _Size>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type **std::__copy_n_a** (istreambuf_iterator< _CharT > __it, _Size __n, _CharT *__result, bool __strict)
- template<typename _Iter>
  constexpr iterator_traits< _Iter >::iterator_category std::__iterator_category (const _Iter &)
- template<typename _Iterator, typename _ReturnType = __conditional_t<__move_if_noexcept_cond <typename iterator_traits<_Iterator>::value_type>::value, _Iterator, move_iterator<_Iterator>>>
  constexpr _ReturnType **std::__make_move_if_noexcept_iterator** (_Iterator __i)
- template<typename _Tp, typename _ReturnType = __conditional_t<__move_if_noexcept_cond<_Tp>::value, const _Tp*, move_iterator<_Tp*>>>
  constexpr _ReturnType **std::__make_move_if_noexcept_iterator** (_Tp *__i)
- template<typename _Iterator>
  constexpr reverse_iterator< _Iterator > **std::__make_reverse_iterator** (_Iterator __i)
- template<typename _CharT, typename _Distance>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, void >::__type **std::advance** (istreambuf_iterator< _CharT > &__i, _Distance __n)
- template<typename _Container>
  constexpr back_insert_iterator< _Container > std::back_inserter (_Container &__x)
- template<typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type **std::copy** (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, ostreambuf_iterator< _CharT > __result)
- template<typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, istreambuf_iterator< _CharT > >::__type **std::find** (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, const _CharT &__val)
- template<typename _Container>
  constexpr front_insert_iterator< _Container > std::front_inserter (_Container &__x)
- template<typename _Container>
  constexpr insert_iterator< _Container > std::inserter (_Container &__x, std::__detail::__range_iter_t< _Container > __i)

- template<typename _Iterator>
  constexpr move_iterator< _Iterator > **std::make_move_iterator** (_Iterator __i)
- template<typename _Iterator>
  constexpr reverse_iterator< _Iterator > std::make_reverse_iterator (_Iterator __i)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() != __y.base() } -> convertible_to<bool>; }
  constexpr bool **std::operator!=** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _Iterator>
  requires requires { { __x.base() + __n } -> same_as<_Iterator>; }
  constexpr move_iterator< _Iterator > **std::operator+** (typename move_iterator< _Iterator >::difference_type ↩
  __n, const move_iterator< _Iterator > &__x)
- template<typename _Iterator>
  constexpr reverse_iterator< _Iterator > **std::operator+** (typename reverse_iterator< _Iterator >::difference_↩
  type __n, const reverse_iterator< _Iterator > &__x)
- template<typename _IteratorL, typename _IteratorR>
  constexpr auto **std::operator-** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &↩
  __y) -> decltype(__x.base() - __y.base())
- template<typename _IteratorL, typename _IteratorR>
  constexpr auto **std::operator-** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y) -> decltype(__y.base() - __x.base())
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() < __y.base() } -> convertible_to<bool>; }
  constexpr bool **std::operator<** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() > __y.base() } -> convertible_to<bool>; }
  constexpr bool **std::operator>** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __y.base() < __x.base() } -> convertible_to<bool>; }
  constexpr bool **std::operator<=** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() >= __y.base() } -> convertible_to<bool>; }
  constexpr bool **std::operator>=** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<three_way_comparable _Iterator>
  constexpr compare_three_way_result_t< _Iterator > **std::operator<=>** (const move_iterator< _Iterator > &↩
  __x, const move_iterator< _Iterator > &__y)
- template<typename _IteratorL, three_way_comparable_with< _IteratorL > _IteratorR>
  constexpr compare_three_way_result_t< _IteratorL, _IteratorR > **std::operator<=>** (const move_iterator< _↩
  IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<three_way_comparable _Iterator>
  constexpr compare_three_way_result_t< _Iterator, _Iterator > **std::operator<=>** (const reverse_iterator< _↩
  Iterator > &__x, const reverse_iterator< _Iterator > &__y)
- template<typename _IteratorL, three_way_comparable_with< _IteratorL > _IteratorR>
  constexpr compare_three_way_result_t< _IteratorL, _IteratorR > **std::operator<=>** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _CharT, typename _Traits>
  bool **std::operator==** (const istreambuf_iterator< _CharT, _Traits > &__a, const istreambuf_iterator< _CharT, _Traits > &__b)

- template<typename _Iterator>
  constexpr bool **std::operator==** (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() == __y.base() } -> convertible_to<bool>; }
  constexpr bool **std::operator==** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _Iterator>
  requires requires { { __x.base() == __y.base() } -> convertible_to<bool>; }
  constexpr bool **std::operator==** (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() == __y.base() } -> convertible_to<bool>; }
  constexpr bool std::operator== (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __y.base() < __x.base() } -> convertible_to<bool>; }
  constexpr bool **std::operator>** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() < __y.base() } -> convertible_to<bool>; }
  constexpr bool **std::operator>** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() < __y.base() } -> convertible_to<bool>; }
  constexpr bool **std::operator>=** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() <= __y.base() } -> convertible_to<bool>; }
  constexpr bool **std::operator>=** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)

Variables

- template<typename _Iterator1, typename _Iterator2>
  constexpr bool **std::disable_sized_sentinel_for**< **move_iterator**< **_Iterator1** >, **move_iterator**< **_Iterator2** > >
- template<typename _Iterator1, typename _Iterator2>
  constexpr bool **std::disable_sized_sentinel_for**< **reverse_iterator**< **_Iterator1** >, **reverse_iterator**< **_Iterator2** > >

3.9.1 Detailed Description

Abstractions for uniform iterating through various underlying types.

3.9.2 Function Documentation

__iterator_category()

```
template<typename _Iter>
iterator_traits< _Iter >::iterator_category std::__iterator_category (
            const _Iter & )  [inline], [constexpr]
```

This function is not a part of the C++ standard but is syntactic sugar for internal library use only.
Referenced by std::basic_string< _CharT >::basic_string(), std::deque< _Tp, polymorphic_allocator< _Tp > >::deque(), std::vector< _Tp, polymorphic_allocator< _Tp > >::vector(), __gnu_debug::__valid_range_aux(), advance(), std::deque< _Tp, polymorp

std::vector< _Tp, polymorphic_allocator< _Tp > >::assign(), distance(), fill_n(), find_end(), find_end(), std::deque< _Tp, polymorphic_al
std::vector< _Tp, polymorphic_allocator< _Tp > >::insert(), partition(), reverse(), rotate(), uninitialized_copy_n(),
unique_copy(), and unique_copy().

back_inserter()

```
template<typename _Container>
back_insert_iterator< _Container > std::back_inserter (
            _Container & __x)  [inline], [nodiscard], [constexpr]
```

Parameters

_↩ _x	A container of arbitrary type.

Returns

An instance of back_insert_iterator working on __x.

This wrapper function helps in creating back_insert_iterator instances. Typing the name of the iterator requires knowing the precise full type of the container, which can be tedious and impedes generic programming. Using this function lets you take advantage of automatic template parameter deduction, making the compiler match the correct types for you.
Referenced by std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const cha
std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const char * >::format(),
regex_replace(), regex_replace(), regex_replace(), and regex_replace().

front_inserter()

```
template<typename _Container>
front_insert_iterator< _Container > std::front_inserter (
            _Container & __x)  [inline], [nodiscard], [constexpr]
```

Parameters

_↩ _x	A container of arbitrary type.

Returns

An instance of front_insert_iterator working on x.

This wrapper function helps in creating front_insert_iterator instances. Typing the name of the iterator requires knowing the precise full type of the container, which can be tedious and impedes generic programming. Using this function lets you take advantage of automatic template parameter deduction, making the compiler match the correct types for you.

inserter()

```
template<typename _Container>
insert_iterator< _Container > std::inserter (
            _Container & __x,
            std::__detail::__range_iter_t< _Container > __i)  [nodiscard], [constexpr]
```

Parameters

_↩ _x	A container of arbitrary type.
_↩ _i	An iterator into the container.

Returns

An instance of insert_iterator working on __x.

This wrapper function helps in creating insert_iterator instances. Typing the name of the iterator requires knowing the precise full type of the container, which can be tedious and impedes generic programming. Using this function lets you take advantage of automatic template parameter deduction, making the compiler match the correct types for you.

make_reverse_iterator()

```
template<typename _Iterator>
reverse_iterator< _Iterator > std::make_reverse_iterator (
            _Iterator __i)  [inline], [constexpr]
```

Generator function for reverse_iterator.

operator==()

```
template<typename _IteratorL, typename _IteratorR>
requires requires { { __x.base() == __y.base() } -> convertible_to<bool>; }
bool std::operator== (
            const reverse_iterator< _IteratorL > & __x,
            const reverse_iterator< _IteratorR > & __y)  [nodiscard], [constexpr]
```

Parameters

_↩ _x	A reverse_iterator.
_↩ _y	A reverse_iterator.

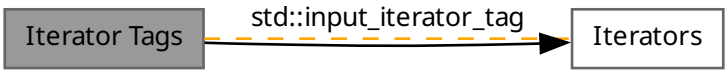
Returns

A simple bool.

Reverse iterators forward comparisons to their underlying base() iterators.
References std::reverse_iterator< _Iterator >::base().

3.9.3 Iterator Tags

Collaboration diagram for Iterator Tags:



Classes

- struct std::bidirectional_iterator_tag
- struct std::contiguous_iterator_tag
- struct std::forward_iterator_tag
- struct std::input_iterator_tag
- struct std::output_iterator_tag
- struct std::random_access_iterator_tag

3.9.3.1 Detailed Description

These are empty types, used to distinguish different iterators. The distinction is not made by what they contain, but simply by what they are. Different underlying algorithms can then be used based on the different operations supported by different iterator types.

3.10 Locales

Namespaces

- namespace std::__detail

Classes

- class std::codecvt< _InternT, _ExternT, _StateT >
- class std::ctype< _CharT >
- class std::ctype< char >
- class std::ctype< wchar_t >
- class std::locale::facet
- class std::locale::id
- class std::locale
- class std::messages< _CharT >
- struct std::messages_base
- class std::money_base

- class std::money_get< _CharT, _InIter >
- class std::money_put< _CharT, _OutIter >
- class std::moneypunct< _CharT, _Intl >
- class std::num_get< _CharT, _InIter >
- class std::num_put< _CharT, _OutIter >
- class std::numpunct< _CharT >
- class std::time_base
- class std::time_get< _CharT, _InIter >
- class std::time_put< _CharT, _OutIter >
- class std::wbuffer_convert< _Codecvt, _Elem, _Tr >
- class std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >

Functions

- template<typename _OutStr, typename _InChar, typename _Codecvt, typename _State, typename _Fn>
  bool **std::__do_str_codecvt** (const _InChar *__first, const _InChar *__last, _OutStr &__outstr, const _Codecvt &__cvt, _State &__state, size_t &__count, _Fn __fn)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _State>
  bool **std::__str_codecvt_in** (const char *__first, const char *__last, basic_string< _CharT, _Traits, _Alloc > &__outstr, const codecvt< _CharT, char, _State > &__cvt)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _State>
  bool **std::__str_codecvt_in** (const char *__first, const char *__last, basic_string< _CharT, _Traits, _Alloc > &__outstr, const codecvt< _CharT, char, _State > &__cvt, _State &__state, size_t &__count)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _State>
  bool **std::__str_codecvt_in_all** (const char *__first, const char *__last, basic_string< _CharT, _Traits, _Alloc > &__outstr, const codecvt< _CharT, char, _State > &__cvt)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _State>
  bool **std::__str_codecvt_out** (const _CharT *__first, const _CharT *__last, basic_string< char, _Traits, _Alloc > &__outstr, const codecvt< _CharT, char, _State > &__cvt)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _State>
  bool **std::__str_codecvt_out** (const _CharT *__first, const _CharT *__last, basic_string< char, _Traits, _Alloc > &__outstr, const codecvt< _CharT, char, _State > &__cvt, _State &__state, size_t &__count)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _State>
  bool **std::__str_codecvt_out_all** (const _CharT *__first, const _CharT *__last, basic_string< char, _Traits, _Alloc > &__outstr, const codecvt< _CharT, char, _State > &__cvt)
- template<typename _Facet>
  bool std::has_facet (const locale &__loc)
- template<typename _Facet>
  const _Facet & std::use_facet (const locale &__loc)

3.10.1 Detailed Description

Classes and functions for internationalization and localization.

3.10.2 Function Documentation

has_facet()

```
template<typename _Facet>
bool std::has_facet (
            const locale & __loc)  [inline], [nodiscard], [noexcept]
```

Test for the presence of a facet.
has_facet tests the locale argument for the presence of the facet type provided as the template parameter. Facets derived from the facet parameter will also return true.

Template Parameters

_Facet	The facet type to test the presence of.

Parameters

__loc	The locale to test.

Returns

true if __loc contains a facet of type _Facet, else false.

use_facet()

```
template<typename _Facet>
const _Facet & std::use_facet (
            const locale & __loc)  [inline], [nodiscard]
```

Return a facet.
use_facet looks for and returns a reference to a facet of type Facet where Facet is the template parameter. If has_facet(locale) is true, there is a suitable facet to return. It throws std::bad_cast if the locale doesn't contain a facet of type Facet.

Template Parameters

_Facet	The facet type to access.

Parameters

__loc	The locale to use.

Returns

Reference to facet of type Facet.

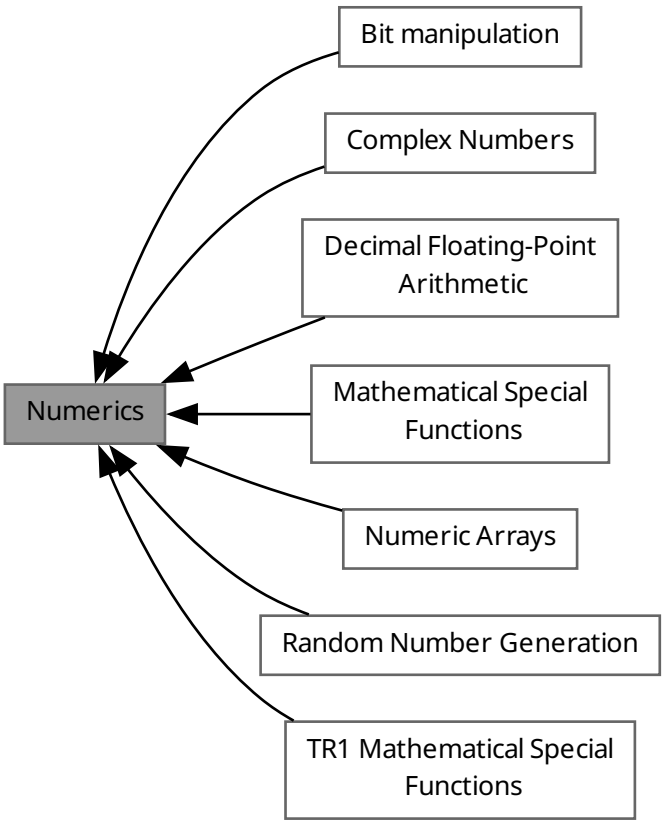
Exceptions

std::bad_cast	if __loc doesn't contain a facet of type _Facet.

Referenced by std::money_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter
std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter
std::time_get< _CharT, _InIter >::do_get_year(), std::money_put< _CharT, _OutIter >::do_put(), std::time_put< _CharT, _OutIter >::do_
std::time_get< _CharT, _InIter >::get(), isalnum(), isalpha(), isblank(), iscntrl(), std::regex_traits< _Ch_type >::isctype(),
isdigit(), isgraph(), islower(), isprint(), ispunct(), isspace(), isupper(), isxdigit(), std::regex_traits< _Ch_type >::lookup_classname(),
std::regex_traits< _CharT >::lookup_classname(), std::regex_traits< _Ch_type >::lookup_collatename(), std::basic_ostream< _CharT,
std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), operator>>(),
operator>>(), std::time_put< _CharT, _OutIter >::put(), tolower(), toupper(), std::regex_traits< _CharT >::transform(),
std::regex_traits< _CharT >::transform_primary(), std::regex_traits< _CharT >::translate_nocase(), and ws().

3.11 Numerics

Collaboration diagram for Numerics:



Topics

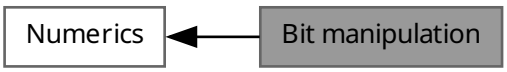
- Bit manipulation
- Complex Numbers
- Decimal Floating-Point Arithmetic
- Mathematical Special Functions
- Numeric Arrays
- Random Number Generation
- TR1 Mathematical Special Functions

3.11.1 Detailed Description

Components for performing numeric operations. Includes support for complex number types, random number generation, numeric (n-at-a-time) arrays, generalized numeric algorithms, and mathematical special functions.

3.11.2 Bit manipulation

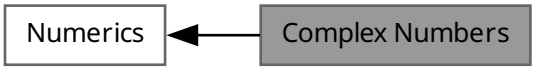
Collaboration diagram for Bit manipulation:



Utilities for examining and manipulating individual bits.

3.11.3 Complex Numbers

Collaboration diagram for Complex Numbers:



Classes

- class std::complex< _Tp >
- class std::complex< double >
- class std::complex< float >
- class std::complex< long double >

Functions

- constexpr **std::complex**< **double** >**::complex** (const complex< long double > &)
- constexpr **std::complex**< **float** >**::complex** (const complex< double > &)
- constexpr **std::complex**< **float** >**::complex** (const complex< long double > &)
- template<typename _Tp>
  _Tp **std::__complex_abs** (const complex< _Tp > &__z)
- template<typename _Tp>
  _Tp **std::__complex_arg** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **std::__complex_cos** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **std::__complex_cosh** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **std::__complex_exp** (const complex< _Tp > &__z)

- template<typename _Tp>
  complex< _Tp > **std::__complex_log** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **std::__complex_pow** (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp>
  complex< _Tp > **std::__complex_pow_unsigned** (complex< _Tp > __x, unsigned __n)
- template<typename _Tp>
  complex< _Tp > **std::__complex_sin** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **std::__complex_sinh** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **std::__complex_sqrt** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **std::__complex_tan** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **std::__complex_tanh** (const complex< _Tp > &__z)
- template<typename _Tp>
  _Tp std::abs (const complex< _Tp > &)
- template<typename _Tp>
  _Tp std::arg (const complex< _Tp > &)
- template<typename _Tp>
  constexpr complex< _Tp > std::conj (const complex< _Tp > &)
- template<typename _Tp>
  std::complex< typename __gnu_cxx::__promote< _Tp >::__type > **std::tr1::conj** (_Tp __x)
- template<typename _Tp>
  std::complex< _Tp > **std::tr1::conj** (const std::complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > std::cos (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > std::cosh (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > std::exp (const complex< _Tp > &)
- template<typename _Tp>
  std::complex< _Tp > std::tr1::fabs (const std::complex< _Tp > &__z)
- template<typename _Tp>
  constexpr _Tp **std::imag** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > std::log (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > std::log10 (const complex< _Tp > &)
- template<typename _Tp>
  _Tp constexpr std::norm (const complex< _Tp > &)
- constexpr complex< _Tp > & std::complex< _Tp >::operator*= (const _Tp &)
- template<typename _Up>
  constexpr complex< _Tp > & std::complex< _Tp >::operator*= (const complex< _Up > &)
- template<typename _Tp>
  constexpr complex< _Tp > std::operator+ (const complex< _Tp > &__x)
- template<typename _Up>
  constexpr complex< _Tp > & std::complex< _Tp >::operator+= (const complex< _Up > &)
- template<typename _Tp>
  constexpr complex< _Tp > std::operator- (const complex< _Tp > &__x)
- template<typename _Up>
  constexpr complex< _Tp > & std::complex< _Tp >::operator-= (const complex< _Up > &)

- constexpr complex< _Tp > & std::complex< _Tp >::operator/= (const _Tp &)
- template<typename _Up>
  constexpr complex< _Tp > & std::complex< _Tp >::operator/= (const complex< _Up > &)
- template<typename _Tp, typename _CharT, class _Traits>
  basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, const complex< _Tp > &__x)
- template ostream & **std::operator**<< (ostream &, const complex< double > &)
- template ostream & **std::operator**<< (ostream &, const complex< float > &)
- template ostream & **std::operator**<< (ostream &, const complex< long double > &)
- template wostream & **std::operator**<< (wostream &, const complex< double > &)
- template wostream & **std::operator**<< (wostream &, const complex< float > &)
- template wostream & **std::operator**<< (wostream &, const complex< long double > &)
- constexpr complex< _Tp > & std::complex< _Tp >::operator= (const _Tp &)
- template<typename _Up>
  constexpr complex< _Tp > & std::complex< _Tp >::operator= (const complex< _Up > &)
- template<typename _Tp, typename _CharT, class _Traits>
  basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, complex< _Tp > &__x)
- template istream & **std::operator**>> (istream &, complex< double > &)
- template istream & **std::operator**>> (istream &, complex< float > &)
- template istream & **std::operator**>> (istream &, complex< long double > &)
- template wistream & **std::operator**>> (wistream &, complex< double > &)
- template wistream & **std::operator**>> (wistream &, complex< float > &)
- template wistream & **std::operator**>> (wistream &, complex< long double > &)
- template<typename _Tp>
  complex< _Tp > std::polar (const _Tp &, const _Tp &=_Tp(0))
- template<typename _Tp, typename _Up>
  std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > **std::tr1::polar** (const _Tp &__rho, const _Up &__theta)
- template<typename _Tp>
  complex< _Tp > std::pow (const _Tp &, const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > std::pow (const complex< _Tp > &, const _Tp &)
- template<typename _Tp>
  complex< _Tp > std::pow (const complex< _Tp > &, const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > std::pow (const complex< _Tp > &, int)
- template<typename _Tp>
  std::complex< _Tp > **std::tr1::pow** (const _Tp &__x, const std::complex< _Tp > &__y)
- template<typename _Tp, typename _Up>
  std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > **std::tr1::pow** (const _Tp &__x, const std::complex< _Up > &__y)
- template<typename _Tp>
  std::complex< _Tp > **std::tr1::pow** (const std::complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp, typename _Up>
  std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::tr1::pow (const std::complex< _Tp > &__x, const _Up &__y)
- template<typename _Tp>
  std::complex< _Tp > **std::tr1::pow** (const std::complex< _Tp > &__x, const std::complex< _Tp > &__y)
- template<typename _Tp, typename _Up>
  std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > **std::tr1::pow** (const std::complex< _Tp > &__x, const std::complex< _Up > &__y)

- template<typename _Tp>
  constexpr _Tp **std::real** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > std::sin (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > std::sinh (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > std::sqrt (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > std::tan (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > std::tanh (const complex< _Tp > &)

- template<typename _Tp>
  constexpr complex< _Tp > std::operator+ (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp>
  constexpr complex< _Tp > std::operator+ (const complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp>
  constexpr complex< _Tp > std::operator+ (const _Tp &__x, const complex< _Tp > &__y)

- template<typename _Tp>
  constexpr complex< _Tp > std::operator- (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp>
  constexpr complex< _Tp > std::operator- (const complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp>
  constexpr complex< _Tp > std::operator- (const _Tp &__x, const complex< _Tp > &__y)

- template<typename _Tp>
  constexpr complex< _Tp > std::operator* (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp>
  constexpr complex< _Tp > std::operator* (const complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp>
  constexpr complex< _Tp > std::operator* (const _Tp &__x, const complex< _Tp > &__y)

- template<typename _Tp>
  constexpr complex< _Tp > std::operator/ (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp>
  constexpr complex< _Tp > std::operator/ (const complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp>
  constexpr complex< _Tp > std::operator/ (const _Tp &__x, const complex< _Tp > &__y)

- template<typename _Tp>
  constexpr bool std::operator== (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp>
  constexpr bool std::operator== (const complex< _Tp > &__x, const _Tp &__y)

3.11.3.1 Detailed Description

Classes and functions for complex numbers.

3.11.3.2 Function Documentation

abs()

```
template<typename _Tp>
_Tp std::abs (
            const complex< _Tp > & __z)  [inline]
```

Return magnitude of *z*.
Referenced by frexp(), std::binomial_distribution< _IntType >::operator()(), and std::poisson_distribution< _IntType >::operator()().

arg()

```
template<typename _Tp>
_Tp std::arg (
            const complex< _Tp > & __z)  [inline]
```

Return phase angle of *z*.

conj()

```
template<typename _Tp>
complex< _Tp > std::conj (
            const complex< _Tp > & __z)  [inline], [constexpr]
```

Return complex conjugate of *z*.

cos()

```
template<typename _Tp>
complex< _Tp > std::cos (
            const complex< _Tp > & __z)  [inline]
```

Return complex cosine of *z*.

cosh()

```
template<typename _Tp>
complex< _Tp > std::cosh (
            const complex< _Tp > & __z)  [inline]
```

Return complex hyperbolic cosine of *z*.

exp()

```
template<typename _Tp>
complex< _Tp > std::exp (
            const complex< _Tp > & __z)  [inline]
```

Return complex base e exponential of *z*.
Referenced by std::shuffle_order_engine< minstd_rand0, 256 >::operator()().

fabs()

```
template<typename _Tp>
std::complex< _Tp > std::tr1::fabs (
            const std::complex< _Tp > & __z)  [inline]
```

fabs(__z) [8.1.8].

log()

```
template<typename _Tp>
complex< _Tp > std::log (
            const complex< _Tp > & __z)  [inline]
```

Return complex natural logarithm of *z*.

Referenced by generate_canonical(), std::binomial_distribution< _IntType >::operator()(), std::gamma_distribution< _RealType >::opera std::normal_distribution< _RealType >::operator()(), std::poisson_distribution< _IntType >::operator()(), std::shuffle_order_engine< min operator<<(), operator<<(), operator<<(), and operator<<().

log10()

```
template<typename _Tp>
complex< _Tp > std::log10 (
            const complex< _Tp > & __z)  [inline]
```

Return complex base 10 logarithm of *z*.

norm()

```
template<typename _Tp>
_Tp constexpr std::norm (
            const complex< _Tp > & __z)  [inline], [constexpr]
```

Return *z* magnitude squared.

operator*() [1/3]

```
template<typename _Tp>
complex< _Tp > std::operator* (
            const _Tp & __x,
            const complex< _Tp > & __y)  [inline], [constexpr]
```

Return new complex value *x* times *y*.

operator*() [2/3]

```
template<typename _Tp>
complex< _Tp > std::operator* (
            const complex< _Tp > & __x,
            const _Tp & __y)  [inline], [constexpr]
```

Return new complex value *x* times *y*.

operator*() [3/3]

```
template<typename _Tp>
complex< _Tp > std::operator* (
            const complex< _Tp > & __x,
            const complex< _Tp > & __y)  [inline], [constexpr]
```

Return new complex value *x* times *y*.

operator*=() [1/2]

```
template<typename _Tp>
complex< _Tp > & std::complex< _Tp >::operator*= (
            const _Tp & __t)  [constexpr]
```

Multiply this complex number by a scalar.

operator*=() [2/2]

```
template<typename _Tp>
template<typename _Up>
complex< _Tp > & std::complex< _Tp >::operator*= (
            const complex< _Up > & __z)  [constexpr]
```

Multiply this complex number by another.

operator+() [1/4]

```
template<typename _Tp>
complex< _Tp > std::operator+ (
            const _Tp & __x,
            const complex< _Tp > & __y)  [inline], [constexpr]
```

Return new complex value *x* plus *y*.

operator+() [2/4]

```
template<typename _Tp>
complex< _Tp > std::operator+ (
            const complex< _Tp > & __x)  [inline], [constexpr]
```

Return *x*.

operator+() [3/4]

```
template<typename _Tp>
complex< _Tp > std::operator+ (
            const complex< _Tp > & __x,
            const _Tp & __y)  [inline], [constexpr]
```

Return new complex value *x* plus *y*.

operator+() [4/4]

```
template<typename _Tp>
complex< _Tp > std::operator+ (
            const complex< _Tp > & __x,
            const complex< _Tp > & __y)  [inline], [constexpr]
```

Return new complex value *x* plus *y*.

operator+=()

```
template<typename _Tp>
template<typename _Up>
complex< _Tp > & std::complex< _Tp >::operator+= (
            const complex< _Up > & __z)  [constexpr]
```

Add another complex number to this one.

operator-() [1/4]

```
template<typename _Tp>
complex< _Tp > std::operator- (
            const _Tp & __x,
            const complex< _Tp > & __y)  [inline], [constexpr]
```

Return new complex value *x* minus *y*.

operator-() [2/4]

```
template<typename _Tp>
complex< _Tp > std::operator- (
            const complex< _Tp > & __x)  [inline], [constexpr]
```

Return complex negation of *x*.

operator-() [3/4]

```
template<typename _Tp>
complex< _Tp > std::operator- (
            const complex< _Tp > & __x,
            const _Tp & __y)  [inline], [constexpr]
```

Return new complex value *x* minus *y*.

operator-() [4/4]

```
template<typename _Tp>
complex< _Tp > std::operator- (
            const complex< _Tp > & __x,
            const complex< _Tp > & __y)  [inline], [constexpr]
```

Return new complex value *x* minus *y*.

operator-=()

```
template<typename _Tp>
template<typename _Up>
complex< _Tp > & std::complex< _Tp >::operator-= (
            const complex< _Up > & __z)  [constexpr]
```

Subtract another complex number from this one.

operator/() [1/3]

```
template<typename _Tp>
complex< _Tp > std::operator/ (
            const _Tp & __x,
            const complex< _Tp > & __y)  [inline], [constexpr]
```

Return new complex value *x* divided by *y*.

operator/() [2/3]

```
template<typename _Tp>
complex< _Tp > std::operator/ (
            const complex< _Tp > & __x,
            const _Tp & __y)  [inline], [constexpr]
```

Return new complex value *x* divided by *y*.

operator/() [3/3]

```
template<typename _Tp>
complex< _Tp > std::operator/ (
            const complex< _Tp > & __x,
            const complex< _Tp > & __y)  [inline], [constexpr]
```

Return new complex value *x* divided by *y*.

operator/=() [1/2]

```
template<typename _Tp>
complex< _Tp > & std::complex< _Tp >::operator/= (
            const _Tp & __t)  [constexpr]
```

Divide this complex number by a scalar.

operator/=() [2/2]

```
template<typename _Tp>
template<typename _Up>
complex< _Tp > & std::complex< _Tp >::operator/= (
            const complex< _Up > & __z)  [constexpr]
```

Divide this complex number by another.

operator<<()

```
template<typename _Tp, typename _CharT, class _Traits>
basic_ostream< _CharT, _Traits > & std::operator<< (
            basic_ostream< _CharT, _Traits > & __os,
            const complex< _Tp > & __x)
```

Insertion operator for complex values.

operator=() [1/2]

```
template<typename _Tp>
complex< _Tp > & std::complex< _Tp >::operator= (
            const _Tp & __t)  [constexpr]
```

Assign a scalar to this complex number.

operator=() [2/2]

```
template<typename _Tp>
template<typename _Up>
complex< _Tp > & std::complex< _Tp >::operator= (
            const complex< _Up > & __z)  [constexpr]
```

Assign another complex number to this one.

operator==() [1/2]

```
template<typename _Tp>
bool std::operator== (
            const complex< _Tp > & __x,
            const _Tp & __y)  [inline], [constexpr]
```

Return true if *x* is equal to *y*.

operator==() [2/2]

```
template<typename _Tp>
bool std::operator== (
            const complex< _Tp > & __x,
            const complex< _Tp > & __y)  [inline], [constexpr]
```

Return true if *x* is equal to *y*.

operator>>()

```
template<typename _Tp, typename _CharT, class _Traits>
basic_istream< _CharT, _Traits > & std::operator>> (
            basic_istream< _CharT, _Traits > & __is,
            complex< _Tp > & __x)
```

Extraction operator for complex values.

polar()

```
template<typename _Tp>
complex< _Tp > std::polar (
            const _Tp & __rho,
            const _Tp & __theta = _Tp(0))  [inline]
```

Return complex with magnitude *rho* and angle *theta*.

pow() [1/5]

```
template<typename _Tp>
complex< _Tp > std::pow (
            const _Tp & __x,
            const complex< _Tp > & __y)  [inline]
```

Return *x* to the *y'th* power.

pow() [2/5]

```
template<typename _Tp>
complex< _Tp > std::pow (
            const complex< _Tp > & __x,
            const _Tp & __y)
```

Return *x* to the *y'th* power.

pow() [3/5]

```
template<typename _Tp>
complex< _Tp > std::pow (
            const complex< _Tp > & __x,
            const complex< _Tp > & __y)  [inline]
```

Return *x* to the *y'th* power.

pow() [4/5]

```
template<typename _Tp>
complex< _Tp > std::pow (
            const complex< _Tp > & __z,
            int __n)  [inline]
```

Return *x* to the *y'th* power.

Referenced by std::gamma_distribution< _RealType >::operator()(), and operator<<().

pow() [5/5]

```
template<typename _Tp, typename _Up>
std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::tr1::pow (
            const std::complex< _Tp > & __x,
            const _Up & __y)  [inline]
```

Additional overloads [8.1.9].

sin()

```
template<typename _Tp>
complex< _Tp > std::sin (
            const complex< _Tp > & __z)  [inline]
```

Return complex sine of *z*.

sinh()

```
template<typename _Tp>
complex< _Tp > std::sinh (
            const complex< _Tp > & __z)  [inline]
```

Return complex hyperbolic sine of *z*.

sqrt()

```
template<typename _Tp>
complex< _Tp > std::sqrt (
            const complex< _Tp > & __z)  [inline]
```

Return complex square root of *z*.
Referenced by std::normal_distribution< _RealType >::operator()(), std::shuffle_order_engine< minstd_rand0, 256 >::operator()(), and std::student_t_distribution< _RealType >::operator()().

tan()

```
template<typename _Tp>
complex< _Tp > std::tan (
            const complex< _Tp > & __z)  [inline]
```

Return complex tangent of *z*.
Referenced by operator<<().

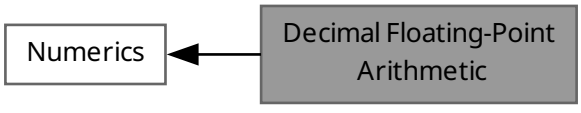
tanh()

```
template<typename _Tp>
complex< _Tp > std::tanh (
            const complex< _Tp > & __z)  [inline]
```

Return complex hyperbolic tangent of *z*.

3.11.4 Decimal Floating-Point Arithmetic

Collaboration diagram for Decimal Floating-Point Arithmetic:



Namespaces

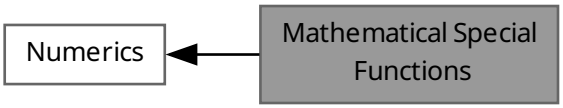
- namespace std::decimal

3.11.4.1 Detailed Description

Classes and functions for decimal floating-point arithmetic.

3.11.5 Mathematical Special Functions

Collaboration diagram for Mathematical Special Functions:



Functions

- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type __gnu_cxx::airy_ai (_Tp __x)
- float __gnu_cxx::airy_aif (float __x)
- long double __gnu_cxx::airy_ail (long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type __gnu_cxx::airy_bi (_Tp __x)
- float __gnu_cxx::airy_bif (float __x)
- long double __gnu_cxx::airy_bil (long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::assoc_laguerre (unsigned int __n, unsigned int __m, _Tp __x)
- float std::assoc_laguerref (unsigned int __n, unsigned int __m, float __x)
- long double std::assoc_laguerrel (unsigned int __n, unsigned int __m, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::assoc_legendre (unsigned int __l, unsigned int __m, _Tp __x)
- float std::assoc_legendref (unsigned int __l, unsigned int __m, float __x)
- long double std::assoc_legendrel (unsigned int __l, unsigned int __m, long double __x)
- template<typename _Tpa, typename _Tpb>
  __gnu_cxx::__promote_2< _Tpa, _Tpb >::__type std::beta (_Tpa __a, _Tpb __b)
- float std::betaf (float __a, float __b)
- long double std::betal (long double __a, long double __b)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::comp_ellint_1 (_Tp __k)
- float std::comp_ellint_1f (float __k)
- long double std::comp_ellint_1l (long double __k)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::comp_ellint_2 (_Tp __k)
- float std::comp_ellint_2f (float __k)
- long double std::comp_ellint_2l (long double __k)

- template<typename _Tp, typename _Tpn>
  __gnu_cxx::__promote_2< _Tp, _Tpn >::__type std::comp_ellint_3 (_Tp __k, _Tpn __nu)
- float std::comp_ellint_3f (float __k, float __nu)
- long double std::comp_ellint_3l (long double __k, long double __nu)
- template<typename _Tpa, typename _Tpc, typename _Tp>
  __gnu_cxx::__promote_3< _Tpa, _Tpc, _Tp >::__type __gnu_cxx::conf_hyperg (_Tpa __a, _Tpc __c, _Tp __x)
- float __gnu_cxx::conf_hypergf (float __a, float __c, float __x)
- long double __gnu_cxx::conf_hypergl (long double __a, long double __c, long double __x)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_bessel_i (_Tpnu __nu, _Tp __x)
- float std::cyl_bessel_if (float __nu, float __x)
- long double std::cyl_bessel_il (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_bessel_j (_Tpnu __nu, _Tp __x)
- float std::cyl_bessel_jf (float __nu, float __x)
- long double std::cyl_bessel_jl (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_bessel_k (_Tpnu __nu, _Tp __x)
- float std::cyl_bessel_kf (float __nu, float __x)
- long double std::cyl_bessel_kl (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_neumann (_Tpnu __nu, _Tp __x)
- float std::cyl_neumannf (float __nu, float __x)
- long double std::cyl_neumannl (long double __nu, long double __x)
- template<typename _Tp, typename _Tpp>
  __gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::ellint_1 (_Tp __k, _Tpp __phi)
- float std::ellint_1f (float __k, float __phi)
- long double std::ellint_1l (long double __k, long double __phi)
- template<typename _Tp, typename _Tpp>
  __gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::ellint_2 (_Tp __k, _Tpp __phi)
- float std::ellint_2f (float __k, float __phi)
- long double std::ellint_2l (long double __k, long double __phi)
- template<typename _Tp, typename _Tpn, typename _Tpp>
  __gnu_cxx::__promote_3< _Tp, _Tpn, _Tpp >::__type std::ellint_3 (_Tp __k, _Tpn __nu, _Tpp __phi)
- float std::ellint_3f (float __k, float __nu, float __phi)
- long double std::ellint_3l (long double __k, long double __nu, long double __phi)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::expint (_Tp __x)
- float std::expintf (float __x)
- long double std::expintl (long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::hermite (unsigned int __n, _Tp __x)
- float std::hermitef (unsigned int __n, float __x)
- long double std::hermitel (unsigned int __n, long double __x)
- template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp>
  __gnu_cxx::__promote_4< _Tpa, _Tpb, _Tpc, _Tp >::__type __gnu_cxx::hyperg (_Tpa __a, _Tpb __b, _Tpc __c, _Tp __x)
- float __gnu_cxx::hypergf (float __a, float __b, float __c, float __x)
- long double __gnu_cxx::hypergl (long double __a, long double __b, long double __c, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::laguerre (unsigned int __n, _Tp __x)
- float std::laguerref (unsigned int __n, float __x)

- long double std::laguerrel (unsigned int __n, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::legendre (unsigned int __l, _Tp __x)
- float std::legendref (unsigned int __l, float __x)
- long double std::legendrel (unsigned int __l, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::riemann_zeta (_Tp __s)
- float std::riemann_zetaf (float __s)
- long double std::riemann_zetal (long double __s)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::sph_bessel (unsigned int __n, _Tp __x)
- float std::sph_besself (unsigned int __n, float __x)
- long double std::sph_bessell (unsigned int __n, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::sph_legendre (unsigned int __l, unsigned int __m, _Tp __theta)
- float std::sph_legendref (unsigned int __l, unsigned int __m, float __theta)
- long double std::sph_legendrel (unsigned int __l, unsigned int __m, long double __theta)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::sph_neumann (unsigned int __n, _Tp __x)
- float std::sph_neumannf (unsigned int __n, float __x)
- long double std::sph_neumannl (unsigned int __n, long double __x)

3.11.5.1 Detailed Description

3.11.5.2 Mathematical Special Functions

A collection of advanced mathematical special functions, defined by ISO/IEC IS 29124 and then added to ISO C++ 2017.

3.11.5.2.1 Introduction and History

The first significant library upgrade on the road to C++2011, TR1, included a set of 23 mathematical functions that significantly extended the standard transcendental functions inherited from C and declared in <cmath>.
Although most components from TR1 were eventually adopted for C++11 these math functions were left behind out of concern for implementability. The math functions were published as a separate international standard IS 29124 - Extensions to the C++ Library to Support Mathematical Special Functions.
For C++17 these functions were incorporated into the main standard.

3.11.5.2.2 Contents

The following functions are implemented in namespace `std`:

- assoc_laguerre - Associated Laguerre functions
- assoc_legendre - Associated Legendre functions
- beta - Beta functions
- comp_ellint_1 - Complete elliptic functions of the first kind
- comp_ellint_2 - Complete elliptic functions of the second kind
- comp_ellint_3 - Complete elliptic functions of the third kind
- cyl_bessel_i - Regular modified cylindrical Bessel functions
- cyl_bessel_j - Cylindrical Bessel functions of the first kind

- cyl_bessel_k - Irregular modified cylindrical Bessel functions
- cyl_neumann - Cylindrical Neumann functions or Cylindrical Bessel functions of the second kind
- ellint_1 - Incomplete elliptic functions of the first kind
- ellint_2 - Incomplete elliptic functions of the second kind
- ellint_3 - Incomplete elliptic functions of the third kind
- expint - The exponential integral
- hermite - Hermite polynomials
- laguerre - Laguerre functions
- legendre - Legendre polynomials
- riemann_zeta - The Riemann zeta function
- sph_bessel - Spherical Bessel functions
- sph_legendre - Spherical Legendre functions
- sph_neumann - Spherical Neumann functions

The hypergeometric functions were stricken from the TR29124 and C++17 versions of this math library because of implementation concerns. However, since they were in the TR1 version and since they are popular we kept them as an extension in namespace `__gnu_cxx`:

- conf_hyperg - Confluent hypergeometric functions
- hyperg - Hypergeometric functions

3.11.5.2.3 Argument Promotion

The arguments suppled to the non-suffixed functions will be promoted according to the following rules:

1. If any argument intended to be floating point is given an integral value That integral value is promoted to double.
2. All floating point arguments are promoted up to the largest floating point precision among them.

3.11.5.2.4 NaN Arguments

If any of the floating point arguments supplied to these functions is invalid or NaN (std::numeric_limits<Tp>::quiet_NaN), the value NaN is returned.

3.11.5.2.5 Implementation

We strive to implement the underlying math with type generic algorithms to the greatest extent possible. In practice, the functions are thin wrappers that dispatch to function templates. Type dependence is controlled with std::numeric_limits and functions thereof.
We don't promote `float` to `double` or `double` to `long double` reflexively. The goal is for `float` functions to operate more quickly, at the cost of `float` accuracy and possibly a smaller domain of validity. Similaryly, `long double` should give you more dynamic range and slightly more pecision than `double` on many systems.

3.11.5.2.6 Testing

These functions have been tested against equivalent implementations from the Gnu Scientific Library, GSL and Boost and the ratio

$$\frac{|f - f_{test}|}{|f_{test}|}$$

is generally found to be within 10^{-15} for 64-bit double on linux-x86_64 systems over most of the ranges of validity.

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

3.11.5.2.7 General Bibliography

See also

Abramowitz and Stegun: Handbook of Mathematical Functions, with Formulas, Graphs, and Mathematical Tables Edited by Milton Abramowitz and Irene A. Stegun, National Bureau of Standards Applied Mathematics Series - 55 Issued June 1964, Tenth Printing, December 1972, with corrections Electronic versions of A&S abound including both pdf and navigable html.

for example http://people.math.sfu.ca/~cbm/aands/

The old A&S has been redone as the NIST Digital Library of Mathematical Functions: http://dlmf.nist.gov/ This version is far more navigable and includes more recent work.

An Atlas of Functions: with Equator, the Atlas Function Calculator 2nd Edition, by Oldham, Keith B., Myland, Jan, Spanier, Jerome

Asymptotics and Special Functions by Frank W. J. Olver, Academic Press, 1974

Numerical Recipes in C, The Art of Scientific Computing, by William H. Press, Second Ed., Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery, Cambridge University Press, 1992

The Special Functions and Their Approximations: Volumes 1 and 2, by Yudell L. Luke, Academic Press, 1969

3.11.5.3 Function Documentation

airy_ai()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type __gnu_cxx::airy_ai (
            _Tp __x)  [inline]
```

Return the Airy function $Ai(x)$ of real argument x.

airy_aif()

```
float __gnu_cxx::airy_aif (
            float __x)  [inline]
```

Return the Airy function $Ai(x)$ of `float` argument x.

airy_ail()

```
long double __gnu_cxx::airy_ail (
            long double __x)  [inline]
```

Return the Airy function $Ai(x)$ of `long double` argument x.

airy_bi()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type __gnu_cxx::airy_bi (
            _Tp __x)  [inline]
```

Return the Airy function $Bi(x)$ of real argument x.

airy_bif()

```
float __gnu_cxx::airy_bif (
            float __x)  [inline]
```

Return the Airy function $Bi(x)$ of `float` argument x.

airy_bil()

```
long double __gnu_cxx::airy_bil (
            long double __x)  [inline]
```

Return the Airy function $Bi(x)$ of `long double` argument x.

assoc_laguerre()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::assoc_laguerre (
            unsigned int __n,
            unsigned int __m,
            _Tp __x)  [inline]
```

Return the associated Laguerre polynomial of nonnegative order `n`, nonnegative degree `m` and real argument `x` : $L_n^m(x)$.

The associated Laguerre function of real degree α, $L_n^\alpha(x)$, is defined by

$$L_n^\alpha(x) = \frac{(\alpha+1)_n}{n!} {}_1F_1(-n; \alpha+1; x)$$

where $(\alpha)_n$ is the Pochhammer symbol and ${}_1F_1(a; c; x)$ is the confluent hypergeometric function.

The associated Laguerre polynomial is defined for integral degree $\alpha = m$ by:

$$L_n^m(x) = (-1)^m \frac{d^m}{dx^m} L_{n+m}(x)$$

where the Laguerre polynomial is defined by:

$$L_n(x) = \frac{e^x}{n!} \frac{d^n}{dx^n}(x^n e^{-x})$$

and $x >= 0$.

See also

laguerre for details of the Laguerre function of degree `n`

Template Parameters

_Tp	The floating-point type of the argument `__x`.

Parameters

_n	The order of the Laguerre function, `__n >= 0`.
_m	The degree of the Laguerre function, `__m >= 0`.
_x	The argument of the Laguerre function, `__x >= 0`.

Exceptions

std::domain_error	if __x < 0.

assoc_laguerref()

```
float std::assoc_laguerref (
            unsigned int __n,
            unsigned int __m,
            float __x)  [inline]
```

Return the associated Laguerre polynomial of order n, degree m: $L_n^m(x)$ for float argument.

See also

assoc_laguerre for more details.

assoc_laguerrel()

```
long double std::assoc_laguerrel (
            unsigned int __n,
            unsigned int __m,
            long double __x)  [inline]
```

Return the associated Laguerre polynomial of order n, degree m: $L_n^m(x)$.

See also

assoc_laguerre for more details.

assoc_legendre()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::assoc_legendre (
            unsigned int __l,
            unsigned int __m,
            _Tp __x)  [inline]
```

Return the associated Legendre function of degree l and order m.
The associated Legendre function is derived from the Legendre function $P_l(x)$ by the Rodrigues formula:

$$P_l^m(x) = (1 - x^2)^{m/2} \frac{d^m}{dx^m} P_l(x)$$

See also

legendre for details of the Legendre function of degree l

Template Parameters

_Tp	The floating-point type of the argument __x.

Parameters

_↩ _*l*	The degree __l >= 0.
_↩ _*m*	The order __m <= l.
_↩ _*x*	The argument, abs(__x) <= 1.

Exceptions

std::domain_error	if abs(__x) > 1.

assoc_legendref()

```
float std::assoc_legendref (
            unsigned int __l,
            unsigned int __m,
            float __x)  [inline]
```

Return the associated Legendre function of degree l and order m for float argument.

See also

assoc_legendre for more details.

assoc_legendrel()

```
long double std::assoc_legendrel (
            unsigned int __l,
            unsigned int __m,
            long double __x)  [inline]
```

Return the associated Legendre function of degree l and order m.

See also

assoc_legendre for more details.

beta()

```
template<typename _Tpa, typename _Tpb>
__gnu_cxx::__promote_2< _Tpa, _Tpb >::__type std::beta (
            _Tpa __a,
            _Tpb __b)  [inline]
```

Return the beta function, $B(a,b)$, for real parameters a, b.
The beta function is defined by

$$B(a,b) = \int_0^1 t^{a-1}(1-t)^{b-1}dt = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}$$

where $a > 0$ and $b > 0$

Template Parameters

_Tpa	The floating-point type of the parameter __a.
_Tpb	The floating-point type of the parameter __b.

Parameters

__a	The first argument of the beta function, __a > 0 .
__b	The second argument of the beta function, __b > 0 .

Exceptions

std::domain_error	if __a < 0 or __b < 0 .

betaf()

```
float std::betaf (
            float __a,
            float __b)  [inline]
```

Return the beta function, $B(a, b)$, for float parameters a, b.

See also

beta for more details.

betal()

```
long double std::betal (
            long double __a,
            long double __b)  [inline]
```

Return the beta function, $B(a, b)$, for long double parameters a, b.

See also

beta for more details.

comp_ellint_1()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::comp_ellint_1 (
            _Tp __k)  [inline]
```

Return the complete elliptic integral of the first kind $K(k)$ for real modulus k.
The complete elliptic integral of the first kind is defined as

$$K(k) = F(k, \pi/2) = \int_0^{\pi/2} \frac{d\theta}{\sqrt{1 - k^2 sin^2\theta}}$$

where $F(k, \phi)$ is the incomplete elliptic integral of the first kind and the modulus $|k| <= 1$.

See also

ellint_1 for details of the incomplete elliptic function of the first kind.

Template Parameters

_Tp	The floating-point type of the modulus __k.

Parameters

_↩ _k	The modulus, abs(__k) <= 1

Exceptions

std::domain_error	if abs(__k) > 1 .

comp_ellint_1f()

```
float std::comp_ellint_1f (
            float __k)  [inline]
```

Return the complete elliptic integral of the first kind $E(k)$ for float modulus k.

See also

comp_ellint_1 for details.

comp_ellint_1l()

```
long double std::comp_ellint_1l (
            long double __k)  [inline]
```

Return the complete elliptic integral of the first kind $E(k)$ for long double modulus k.

See also

comp_ellint_1 for details.

comp_ellint_2()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::comp_ellint_2 (
            _Tp __k)  [inline]
```

Return the complete elliptic integral of the second kind $E(k)$ for real modulus k.
The complete elliptic integral of the second kind is defined as

$$E(k) = E(k, \pi/2) = \int_0^{\pi/2} \sqrt{1 - k^2 sin^2\theta}$$

where $E(k, \phi)$ is the incomplete elliptic integral of the second kind and the modulus $|k| <= 1$.

See also

ellint_2 for details of the incomplete elliptic function of the second kind.

Template Parameters

_Tp	The floating-point type of the modulus __k.

Parameters

_↩ _*k*	The modulus, `abs(__k) <= 1`

Exceptions

std::domain_error	if `abs(__k) > 1`.

comp_ellint_2f()

```
float std::comp_ellint_2f (
            float __k)  [inline]
```

Return the complete elliptic integral of the second kind $E(k)$ for `float` modulus `k`.

See also

comp_ellint_2 for details.

comp_ellint_2l()

```
long double std::comp_ellint_2l (
            long double __k)  [inline]
```

Return the complete elliptic integral of the second kind $E(k)$ for long double modulus `k`.

See also

comp_ellint_2 for details.

comp_ellint_3()

```
template<typename _Tp, typename _Tpn>
__gnu_cxx::__promote_2< _Tp, _Tpn >::__type std::comp_ellint_3 (
            _Tp __k,
            _Tpn __nu)  [inline]
```

Return the complete elliptic integral of the third kind $\Pi(k,\nu) = \Pi(k,\nu,\pi/2)$ for real modulus `k`.
The complete elliptic integral of the third kind is defined as

$$\Pi(k,\nu) = \Pi(k,\nu,\pi/2) = \int_0^{\pi/2} \frac{d\theta}{(1-\nu\sin^2\theta)\sqrt{1-k^2\sin^2\theta}}$$

where $\Pi(k,\nu,\phi)$ is the incomplete elliptic integral of the second kind and the modulus $|k| <= 1$.

See also

ellint_3 for details of the incomplete elliptic function of the third kind.

Template Parameters

_Tp	The floating-point type of the modulus `__k`.
_Tpn	The floating-point type of the argument `__nu`.

Parameters

__k	The modulus, abs(__k) <= 1
__nu	The argument

Exceptions

std::domain_error	if abs(__k) > 1.

comp_ellint_3f()

```
float std::comp_ellint_3f (
            float __k,
            float __nu)  [inline]
```

Return the complete elliptic integral of the third kind $\Pi(k,\nu)$ for float modulus k.

See also

comp_ellint_3 for details.

comp_ellint_3l()

```
long double std::comp_ellint_3l (
            long double __k,
            long double __nu)  [inline]
```

Return the complete elliptic integral of the third kind $\Pi(k,\nu)$ for long double modulus k.

See also

comp_ellint_3 for details.

conf_hyperg()

```
template<typename _Tpa, typename _Tpc, typename _Tp>
__gnu_cxx::__promote_3< _Tpa, _Tpc, _Tp >::__type __gnu_cxx::conf_hyperg (
            _Tpa __a,
            _Tpc __c,
            _Tp __x)  [inline]
```

Return the confluent hypergeometric function ${}_1F_1(a;c;x)$ of real numeratorial parameter a, denominatorial parameter c, and argument x.
The confluent hypergeometric function is defined by

$$ {}_1F_1(a;c;x) = \sum_{n=0}^{\infty} \frac{(a)_n x^n}{(c)_n n!} $$

where the Pochhammer symbol is $(x)_k = (x)(x+1)...(x+k-1)$, $(x)_0 = 1$

Parameters

_↩ _a	The numeratorial parameter
_↩ _c	The denominatorial parameter

_↩ _x	The argument

conf_hypergf()

```
float __gnu_cxx::conf_hypergf (
             float __a,
             float __c,
             float __x)  [inline]
```

Return the confluent hypergeometric function ${}_1F_1(a;c;x)$ of `float` numeratorial parameter `a`, denominatorial parameter `c`, and argument `x`.

See also

conf_hyperg for details.

conf_hypergl()

```
long double __gnu_cxx::conf_hypergl (
             long double __a,
             long double __c,
             long double __x)  [inline]
```

Return the confluent hypergeometric function ${}_1F_1(a;c;x)$ of `long double` numeratorial parameter `a`, denominatorial parameter `c`, and argument `x`.

See also

conf_hyperg for details.

cyl_bessel_i()

```
template<typename _Tpnu, typename _Tp>
__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_bessel_i (
             _Tpnu __nu,
             _Tp __x)  [inline]
```

Return the regular modified Bessel function $I_\nu(x)$ for real order ν and argument $x >= 0$.
The regular modified cylindrical Bessel function is:

$$I_\nu(x) = i^{-\nu} J_\nu(ix) = \sum_{k=0}^{\infty} \frac{(x/2)^{\nu+2k}}{k!\Gamma(\nu+k+1)}$$

Template Parameters

_Tpnu	The floating-point type of the order `__nu`.
_Tp	The floating-point type of the argument `__x`.

Parameters

__nu	The order
__x	The argument, `__x >= 0`

Exceptions

std::domain_error	if __x < 0 .

cyl_bessel_if()

```
float std::cyl_bessel_if (
            float __nu,
            float __x)  [inline]
```

Return the regular modified Bessel function $I_\nu(x)$ for `float` order ν and argument $x >= 0$.

See also

cyl_bessel_i for setails.

cyl_bessel_il()

```
long double std::cyl_bessel_il (
            long double __nu,
            long double __x)  [inline]
```

Return the regular modified Bessel function $I_\nu(x)$ for `long double` order ν and argument $x >= 0$.

See also

cyl_bessel_i for setails.

cyl_bessel_j()

```
template<typename _Tpnu, typename _Tp>
__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_bessel_j (
            _Tpnu __nu,
            _Tp __x)  [inline]
```

Return the Bessel function $J_\nu(x)$ of real order ν and argument $x >= 0$.
The cylindrical Bessel function is:

$$J_\nu(x) = \sum_{k=0}^{\infty} \frac{(-1)^k (x/2)^{\nu+2k}}{k!\Gamma(\nu+k+1)}$$

Template Parameters

_Tpnu	The floating-point type of the order __nu.
_Tp	The floating-point type of the argument __x.

Parameters

__nu	The order
__x	The argument, __x >= 0

Exceptions

std::domain_error	if __x < 0 .

cyl_bessel_jf()

```
float std::cyl_bessel_jf (
            float __nu,
            float __x)  [inline]
```

Return the Bessel function of the first kind $J_\nu(x)$ for `float` order ν and argument $x >= 0$.

See also

cyl_bessel_j for setails.

cyl_bessel_jl()

```
long double std::cyl_bessel_jl (
            long double __nu,
            long double __x)  [inline]
```

Return the Bessel function of the first kind $J_\nu(x)$ for `long double` order ν and argument $x >= 0$.

See also

cyl_bessel_j for setails.

cyl_bessel_k()

```
template<typename _Tpnu, typename _Tp>
__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_bessel_k (
            _Tpnu __nu,
            _Tp __x)  [inline]
```

Return the irregular modified Bessel function $K_\nu(x)$ of real order ν and argument x.
The irregular modified Bessel function is defined by:

$$K_\nu(x) = \frac{\pi}{2}\frac{I_{-\nu}(x) - I_\nu(x)}{\sin\nu\pi}$$

where for integral $\nu = n$ a limit is taken: $lim_{\nu\to n}$. For negative argument we have simply:

$$K_{-\nu}(x) = K_\nu(x)$$

Template Parameters

_Tpnu	The floating-point type of the order `__nu`.
_Tp	The floating-point type of the argument `__x`.

Parameters

__nu	The order
__x	The argument, `__x >= 0`

Exceptions

std::domain_error	if `__x < 0` .

cyl_bessel_kf()

```
float std::cyl_bessel_kf (
            float __nu,
            float __x)  [inline]
```

Return the irregular modified Bessel function $K_\nu(x)$ for `float` order ν and argument $x >= 0$.

See also

cyl_bessel_k for setails.

cyl_bessel_kl()

```
long double std::cyl_bessel_kl (
            long double __nu,
            long double __x)  [inline]
```

Return the irregular modified Bessel function $K_\nu(x)$ for `long double` order ν and argument $x >= 0$.

See also

cyl_bessel_k for setails.

cyl_neumann()

```
template<typename _Tpnu, typename _Tp>
__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_neumann (
            _Tpnu __nu,
            _Tp __x)  [inline]
```

Return the Neumann function $N_\nu(x)$ of real order ν and argument $x >= 0$.
The Neumann function is defined by:

$$N_\nu(x) = \frac{J_\nu(x)\cos\nu\pi - J_{-\nu}(x)}{\sin\nu\pi}$$

where $x >= 0$ and for integral order $\nu = n$ a limit is taken: $lim_{\nu\to n}$.

Template Parameters

_Tpnu	The floating-point type of the order __nu.
_Tp	The floating-point type of the argument __x.

Parameters

__nu	The order
__x	The argument, __x >= 0

Exceptions

std::domain_error	if __x < 0 .

cyl_neumannf()

```
float std::cyl_neumannf (
            float __nu,
            float __x)  [inline]
```

Return the Neumann function $N_\nu(x)$ of `float` order ν and argument x.

See also

cyl_neumann for setails.

cyl_neumannl()

```
long double std::cyl_neumannl (
            long double __nu,
            long double __x)  [inline]
```

Return the Neumann function $N_\nu(x)$ of `long double` order ν and argument x.

See also

cyl_neumann for setails.

ellint_1()

```
template<typename _Tp, typename _Tpp>
__gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::ellint_1 (
            _Tp __k,
            _Tpp __phi)  [inline]
```

Return the incomplete elliptic integral of the first kind $F(k,\phi)$ for `real` modulus k and angle ϕ.
The incomplete elliptic integral of the first kind is defined as

$$F(k,\phi) = \int_0^\phi \frac{d\theta}{\sqrt{1 - k^2 sin^2\theta}}$$

For $\phi = \pi/2$ this becomes the complete elliptic integral of the first kind, $K(k)$.

See also

comp_ellint_1.

Template Parameters

_Tp	The floating-point type of the modulus `__k`.
_Tpp	The floating-point type of the angle `__phi`.

Parameters

__k	The modulus, `abs(__k) <= 1`
__phi	The integral limit argument in radians

Exceptions

std::domain_error	if `abs(__k) > 1` .

ellint_1f()

```
float std::ellint_1f (
            float __k,
            float __phi)  [inline]
```

Return the incomplete elliptic integral of the first kind $E(k, \phi)$ for `float` modulus k and angle ϕ.

See also

ellint_1 for details.

ellint_1l()

```
long double std::ellint_1l (
            long double __k,
            long double __phi)  [inline]
```

Return the incomplete elliptic integral of the first kind $E(k, \phi)$ for `long double` modulus k and angle ϕ.

See also

ellint_1 for details.

ellint_2()

```
template<typename _Tp, typename _Tpp>
__gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::ellint_2 (
            _Tp __k,
            _Tpp __phi)  [inline]
```

Return the incomplete elliptic integral of the second kind $E(k, \phi)$.
The incomplete elliptic integral of the second kind is defined as

$$E(k, \phi) = \int_0^{\phi} \sqrt{1 - k^2 sin^2\theta}$$

For $\phi = \pi/2$ this becomes the complete elliptic integral of the second kind, $E(k)$.

See also

comp_ellint_2.

Template Parameters

_Tp	The floating-point type of the modulus `__k`.
_Tpp	The floating-point type of the angle `__phi`.

Parameters

__k	The modulus, `abs(__k) <= 1`
__phi	The integral limit argument in radians

Returns

The elliptic function of the second kind.

Exceptions

std::domain_error	if `abs(__k) > 1 .`

ellint_2f()

```
float std::ellint_2f (
            float __k,
            float __phi)  [inline]
```

Return the incomplete elliptic integral of the second kind $E(k, \phi)$ for `float` argument.

See also

ellint_2 for details.

ellint_2l()

```
long double std::ellint_2l (
            long double __k,
            long double __phi)  [inline]
```

Return the incomplete elliptic integral of the second kind $E(k, \phi)$.

See also

ellint_2 for details.

ellint_3()

```
template<typename _Tp, typename _Tpn, typename _Tpp>
__gnu_cxx::__promote_3< _Tp, _Tpn, _Tpp >::__type std::ellint_3 (
            _Tp __k,
            _Tpn __nu,
            _Tpp __phi)  [inline]
```

Return the incomplete elliptic integral of the third kind $\Pi(k, \nu, \phi)$.
The incomplete elliptic integral of the third kind is defined by:

$$\Pi(k, \nu, \phi) = \int_0^\phi \frac{d\theta}{(1 - \nu \sin^2\theta)\sqrt{1 - k^2 \sin^2\theta}}$$

For $\phi = \pi/2$ this becomes the complete elliptic integral of the third kind, $\Pi(k, \nu)$.

See also

comp_ellint_3.

Template Parameters

_Tp	The floating-point type of the modulus `__k`.
_Tpn	The floating-point type of the argument `__nu`.
_Tpp	The floating-point type of the angle `__phi`.

Parameters

__k	The modulus, `abs(__k) <= 1`
__nu	The second argument
__phi	The integral limit argument in radians

Returns

The elliptic function of the third kind.

Exceptions

std::domain_error	if `abs(__k) > 1` .

ellint_3f()

```
float std::ellint_3f (
            float __k,
            float __nu,
            float __phi)  [inline]
```

Return the incomplete elliptic integral of the third kind $\Pi(k, \nu, \phi)$ for `float` argument.

See also

ellint_3 for details.

ellint_3l()

```
long double std::ellint_3l (
            long double __k,
            long double __nu,
            long double __phi)  [inline]
```

Return the incomplete elliptic integral of the third kind $\Pi(k, \nu, \phi)$.

See also

ellint_3 for details.

expint()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::expint (
            _Tp __x)  [inline]
```

Return the exponential integral $Ei(x)$ for `real` argument `x`.
The exponential integral is given by

$$Ei(x) = -\int_{-x}^{\infty} \frac{e^t}{t} dt$$

Template Parameters

_Tp	The floating-point type of the argument __x.

Parameters

_↩ _x	The argument of the exponential integral function.

expintf()

```
float std::expintf (
            float __x)  [inline]
```

Return the exponential integral $Ei(x)$ for `float` argument `x`.

See also

expint for details.

expintl()

```
long double std::expintl (
            long double __x)  [inline]
```

Return the exponential integral $Ei(x)$ for `long double` argument `x`.

See also

expint for details.

hermite()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::hermite (
            unsigned int __n,
            _Tp __x)  [inline]
```

Return the Hermite polynomial $H_n(x)$ of order n and `real` argument `x`.
The Hermite polynomial is defined by:

$$H_n(x) = (-1)^n e^{x^2} \frac{d^n}{dx^n} e^{-x^2}$$

The Hermite polynomial obeys a reflection formula:

$$H_n(-x) = (-1)^n H_n(x)$$

Template Parameters

_Tp	The floating-point type of the argument __x.

Parameters

_↩ _n	The order
_↩ _x	The argument

hermitef()

```
float std::hermitef (
            unsigned int __n,
            float __x)  [inline]
```

Return the Hermite polynomial $H_n(x)$ of nonnegative order n and float argument x.

See also

hermite for details.

hermitel()

```
long double std::hermitel (
            unsigned int __n,
            long double __x)  [inline]
```

Return the Hermite polynomial $H_n(x)$ of nonnegative order n and long double argument x.

See also

hermite for details.

hyperg()

```
template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp>
__gnu_cxx::__promote_4< _Tpa, _Tpb, _Tpc, _Tp >::__type __gnu_cxx::hyperg (
            _Tpa __a,
            _Tpb __b,
            _Tpc __c,
            _Tp __x)  [inline]
```

Return the hypergeometric function ${}_2F_1(a,b;c;x)$ of real numeratorial parameters a and b, denominatorial parameter c, and argument x.
The hypergeometric function is defined by

$$ {}_2F_1(a;c;x) = \sum_{n=0}^{\infty} \frac{(a)_n (b)_n x^n}{(c)_n n!} $$

where the Pochhammer symbol is $(x)_k = (x)(x+1)...(x+k-1)$, $(x)_0 = 1$

Parameters

_↩ _a	The first numeratorial parameter
_↩ _b	The second numeratorial parameter
_↩ _c	The denominatorial parameter
_↩ _x	The argument

hypergf()

```
float __gnu_cxx::hypergf (
            float __a,
```

```
            float __b,
            float __c,
            float __x)  [inline]
```

Return the hypergeometric function ${}_2F_1(a,b;c;x)$ of @ float numeratorial parameters a and b, denominatorial parameter c, and argument x.

See also

hyperg for details.

hypergl()

```
long double __gnu_cxx::hypergl (
            long double __a,
            long double __b,
            long double __c,
            long double __x)  [inline]
```

Return the hypergeometric function ${}_2F_1(a,b;c;x)$ of long double numeratorial parameters a and b, denominatorial parameter c, and argument x.

See also

hyperg for details.

laguerre()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::laguerre (
            unsigned int __n,
            _Tp __x)  [inline]
```

Returns the Laguerre polynomial $L_n(x)$ of nonnegative degree n and real argument $x >= 0$.
The Laguerre polynomial is defined by:

$$L_n(x) = \frac{e^x}{n!}\frac{d^n}{dx^n}(x^n e^{-x})$$

Template Parameters

_Tp	The floating-point type of the argument __x.

Parameters

_n	The nonnegative order
_x	The argument __x >= 0

Exceptions

std::domain_error	if __x < 0 .

laguerref()

```
float std::laguerref (
            unsigned int __n,
            float __x)  [inline]
```

Returns the Laguerre polynomial $L_n(x)$ of nonnegative degree `n` and `float` argument $x >= 0$.

See also

laguerre for more details.

laguerrel()

```
long double std::laguerrel (
            unsigned int __n,
            long double __x)  [inline]
```

Returns the Laguerre polynomial $L_n(x)$ of nonnegative degree `n` and `long double` argument $x >= 0$.

See also

laguerre for more details.

legendre()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::legendre (
            unsigned int __l,
            _Tp __x)  [inline]
```

Return the Legendre polynomial $P_l(x)$ of nonnegative degree l and real argument $|x| <= 0$.
The Legendre function of order l and argument x, $P_l(x)$, is defined by:

$$P_l(x) = \frac{1}{2^l l!}\frac{d^l}{dx^l}(x^2 - 1)^l$$

Template Parameters

_Tp	The floating-point type of the argument __x.

Parameters

_↩ _l	The degree $l >= 0$
_↩ _x	The argument `abs(__x)` <= 1

Exceptions

std::domain_error	if `abs(__x)` > 1

legendref()

```
float std::legendref (
            unsigned int __l,
            float __x)  [inline]
```

Return the Legendre polynomial $P_l(x)$ of nonnegative degree l and `float` argument $|x| <= 0$.

See also

legendre for more details.

legendrel()

```
long double std::legendrel (
            unsigned int __l,
            long double __x)  [inline]
```

Return the Legendre polynomial $P_l(x)$ of nonnegative degree l and `long double` argument $|x| <= 0$.

See also

legendre for more details.

riemann_zeta()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::riemann_zeta (
            _Tp __s)  [inline]
```

Return the Riemann zeta function $\zeta(s)$ for real argument s.
The Riemann zeta function is defined by:

$$\zeta(s) = \sum_{k=1}^{\infty} k^{-s} \text{ for } s > 1$$

and

$$\zeta(s) = \frac{1}{1-2^{1-s}} \sum_{k=1}^{\infty} (-1)^{k-1} k^{-s} \text{ for } 0 <= s <= 1$$

For s < 1 use the reflection formula:

$$\zeta(s) = 2^s \pi^{s-1} \sin(\frac{\pi s}{2}) \Gamma(1-s) \zeta(1-s)$$

Template Parameters

_Tp	The floating-point type of the argument __s.

Parameters

_↩ *_s*	The argument `s != 1`

riemann_zetaf()

```
float std::riemann_zetaf (
            float __s)  [inline]
```

Return the Riemann zeta function $\zeta(s)$ for `float` argument s.

See also

riemann_zeta for more details.

riemann_zetal()

```
long double std::riemann_zetal (
            long double __s)  [inline]
```

Return the Riemann zeta function $\zeta(s)$ for `long double` argument s.

See also

riemann_zeta for more details.

sph_bessel()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::sph_bessel (
            unsigned int __n,
            _Tp __x)  [inline]
```

Return the spherical Bessel function $j_n(x)$ of nonnegative order n and real argument $x >= 0$.
The spherical Bessel function is defined by:

$$j_n(x) = \left(\frac{\pi}{2x}\right)^{1/2} J_{n+1/2}(x)$$

Template Parameters

_Tp	The floating-point type of the argument __x.

Parameters

_n	The integral order `n >= 0`
_x	The real argument `x >= 0`

Exceptions

std::domain_error	if `__x < 0` .

sph_besself()

```
float std::sph_besself (
            unsigned int __n,
            float __x)  [inline]
```

Return the spherical Bessel function $j_n(x)$ of nonnegative order n and `float` argument $x >= 0$.

See also

sph_bessel for more details.

sph_bessell()

```
long double std::sph_bessell (
            unsigned int __n,
            long double __x)  [inline]
```

Return the spherical Bessel function $j_n(x)$ of nonnegative order n and `long double` argument $x >= 0$.

See also

sph_bessel for more details.

sph_legendre()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::sph_legendre (
            unsigned int __l,
            unsigned int __m,
            _Tp __theta)  [inline]
```

Return the spherical Legendre function of nonnegative integral degree `l` and order `m` and real angle θ in radians. The spherical Legendre function is defined by

$$Y_l^m(\theta,\phi) = (-1)^m[\frac{(2l+1)}{4\pi}\frac{(l-m)!}{(l+m)!}]P_l^m(\cos\theta)\exp^{im\phi}$$

Template Parameters

_Tp	The floating-point type of the angle `__theta`.

Parameters

__l	The order `__l >= 0`
__m	The degree `__m >= 0` and `__m <= __l`
__theta	The radian polar angle argument

sph_legendref()

```
float std::sph_legendref (
            unsigned int __l,
            unsigned int __m,
            float __theta)  [inline]
```

Return the spherical Legendre function of nonnegative integral degree `l` and order `m` and float angle θ in radians.

See also

sph_legendre for details.

sph_legendrel()

```
long double std::sph_legendrel (
            unsigned int __l,
            unsigned int __m,
            long double __theta)  [inline]
```

Return the spherical Legendre function of nonnegative integral degree `l` and order `m` and `long double` angle θ in radians.

See also

sph_legendre for details.

sph_neumann()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::sph_neumann (
            unsigned int __n,
            _Tp __x)  [inline]
```

Return the spherical Neumann function of integral order $n >= 0$ and real argument $x >= 0$.
The spherical Neumann function is defined by

$$n_n(x) = \left(\frac{\pi}{2x}\right)^{1/2} N_{n+1/2}(x)$$

Template Parameters

_Tp	The floating-point type of the argument __x.

Parameters

_n	The integral order `n >= 0`
_x	The real argument __x >= 0

Exceptions

std::domain_error	if __x < 0 .

sph_neumannf()

```
float std::sph_neumannf (
            unsigned int __n,
            float __x)  [inline]
```

Return the spherical Neumann function of integral order $n >= 0$ and `float` argument $x >= 0$.

See also

sph_neumann for details.

sph_neumannl()

```
long double std::sph_neumannl (
            unsigned int __n,
            long double __x)  [inline]
```

Return the spherical Neumann function of integral order $n >= 0$ and long double $x >= 0$.

See also

sph_neumann for details.

3.11.6 Numeric Arrays

Collaboration diagram for Numeric Arrays:

Classes

- class std::gslice
- class std::gslice_array< _Tp >
- class std::indirect_array< _Tp >
- class std::mask_array< _Tp >
- class std::slice
- class std::slice_array< _Tp >
- class std::valarray< _Tp >

Functions

- std::gslice::gslice ()
- std::gslice::gslice (const gslice &)
- std::gslice::gslice (size_t __o, const valarray< size_t > &__l, const valarray< size_t > &__s)
- std::gslice_array< _Tp >::gslice_array (const gslice_array &)
- std::indirect_array< _Tp >::indirect_array (const indirect_array &)
- std::mask_array< _Tp >::mask_array (const mask_array &)
- std::slice::slice ()
- std::slice::slice (size_t __o, size_t __d, size_t __s)
- std::slice_array< _Tp >::slice_array (const slice_array &)
- std::valarray< _Tp >::valarray () noexcept
- template<class _Dom>
  std::valarray< _Tp >::valarray (const _Expr< _Dom, _Tp > &__e)
- std::valarray< _Tp >::valarray (const _Tp &, size_t)
- template<typename _Tp>
  std::valarray< _Tp >::valarray (const _Tp *__restrict__ __p, size_t __n)
- std::valarray< _Tp >::valarray (const gslice_array< _Tp > &)
- std::valarray< _Tp >::valarray (const indirect_array< _Tp > &)

- std::valarray< _Tp >::valarray (const mask_array< _Tp > &)
- std::valarray< _Tp >::valarray (const slice_array< _Tp > &)
- std::valarray< _Tp >::valarray (const valarray &)
- std::valarray< _Tp >::valarray (initializer_list< _Tp >)
- std::valarray< _Tp >::valarray (size_t)
- std::valarray< _Tp >::valarray (valarray &&) noexcept
- std::gslice::~gslice ()
- _Expr< _ValFunClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::apply (_Tp __func(_Tp)) const
- _Expr< _RefFunClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::apply (_Tp __func(const _Tp &)) const
- template<class _Tp>
  const _Tp * std::begin (const valarray< _Tp > &__va) noexcept
- template<class _Tp>
  _Tp * std::begin (valarray< _Tp > &__va) noexcept
- valarray< _Tp > std::valarray< _Tp >::cshift (int __n) const
- template<class _Tp>
  const _Tp * std::end (const valarray< _Tp > &__va) noexcept
- template<class _Tp>
  _Tp * std::end (valarray< _Tp > &__va) noexcept
- _Tp std::valarray< _Tp >::max () const
- _Tp std::valarray< _Tp >::min () const
- gslice & std::gslice::operator= (const gslice &)
- template<class _Dom>
  void **std::gslice_array< _Tp >::operator=** (const _Expr< _Dom, _Tp > &) const
- void std::gslice_array< _Tp >::operator= (const _Tp &) const
- gslice_array & std::gslice_array< _Tp >::operator= (const gslice_array &)
- void std::gslice_array< _Tp >::operator= (const valarray< _Tp > &) const
- template<class _Dom>
  void **std::indirect_array< _Tp >::operator=** (const _Expr< _Dom, _Tp > &) const
- void std::indirect_array< _Tp >::operator= (const _Tp &) const
- indirect_array & std::indirect_array< _Tp >::operator= (const indirect_array &)
- void std::indirect_array< _Tp >::operator= (const valarray< _Tp > &) const
- template<class _Ex>
  void **std::mask_array< _Tp >::operator=** (const _Expr< _Ex, _Tp > &__e) const
- void std::mask_array< _Tp >::operator= (const _Tp &) const
- mask_array & std::mask_array< _Tp >::operator= (const mask_array &)
- void **std::mask_array< _Tp >::operator=** (const valarray< _Tp > &) const
- template<class _Dom>
  void **std::slice_array< _Tp >::operator=** (const _Expr< _Dom, _Tp > &) const
- void std::slice_array< _Tp >::operator= (const _Tp &) const
- slice_array & std::slice_array< _Tp >::operator= (const slice_array &)
- void std::slice_array< _Tp >::operator= (const valarray< _Tp > &) const
- template<class _Dom>
  valarray< _Tp > & **std::valarray< _Tp >::operator=** (const _Expr< _Dom, _Tp > &)
- valarray< _Tp > & std::valarray< _Tp >::operator= (const _Tp &__t)
- valarray< _Tp > & std::valarray< _Tp >::operator= (const gslice_array< _Tp > &__ga)
- valarray< _Tp > & std::valarray< _Tp >::operator= (const indirect_array< _Tp > &__ia)
- valarray< _Tp > & std::valarray< _Tp >::operator= (const mask_array< _Tp > &__ma)
- valarray< _Tp > & std::valarray< _Tp >::operator= (const slice_array< _Tp > &__sa)
- valarray< _Tp > & std::valarray< _Tp >::operator= (const valarray< _Tp > &__v)
- valarray & std::valarray< _Tp >::operator= (initializer_list< _Tp > __l)
- valarray< _Tp > & std::valarray< _Tp >::operator= (valarray< _Tp > &&__v) noexcept

- gslice_array< _Tp > std::valarray< _Tp >::operator[ ] (const gslice &__s)
- _Expr< _GClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::operator[ ] (const gslice &__s) const
- mask_array< _Tp > std::valarray< _Tp >::operator[ ] (const valarray< bool > &__m)
- valarray< _Tp > std::valarray< _Tp >::operator[ ] (const valarray< bool > &__m) const
- indirect_array< _Tp > std::valarray< _Tp >::operator[ ] (const valarray< size_t > &__i)
- _Expr< _IClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::operator[ ] (const valarray< size_t > &__i) const
- _Tp & std::valarray< _Tp >::operator[ ] (size_t __i) noexcept
- const _Tp & **std::valarray< _Tp >::operator[ ]** (size_t) const noexcept
- slice_array< _Tp > std::valarray< _Tp >::operator[ ] (slice __s)
- _Expr< _SClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::operator[ ] (slice __s) const
- void std::valarray< _Tp >::resize (size_t __size, _Tp __c=_Tp())
- valarray< _Tp > std::valarray< _Tp >::shift (int __n) const
- valarray< size_t > std::gslice::size () const
- size_t std::slice::size () const
- size_t std::valarray< _Tp >::size () const
- size_t std::gslice::start () const
- size_t std::slice::start () const
- valarray< size_t > std::gslice::stride () const
- size_t std::slice::stride () const
- _Tp std::valarray< _Tp >::sum () const
- void std::valarray< _Tp >::swap (valarray< _Tp > &__v) noexcept
- template<typename _Tp, size_t _Nm>
  std::valarray (const _Tp(&)[_Nm], size_t) -> valarray< _Tp >

3.11.6.1 Detailed Description

Classes and functions for representing and manipulating arrays of elements.

3.11.6.2 Function Documentation

gslice() [1/3]

```
std::gslice::gslice ()  [inline]
```

Construct an empty slice.
References gslice().
Referenced by gslice(), gslice(), gslice(), and operator=().

gslice() [2/3]

```
std::gslice::gslice (
            const gslice & __g)  [inline]
```

Copy constructor.
References gslice().

gslice() [3/3]

```
std::gslice::gslice (
            size_t __o,
            const valarray< size_t > & __l,
            const valarray< size_t > & __s)  [inline]
```

Construct a slice.
Constructs a slice with as many dimensions as the length of the *l* and *s* arrays.

Parameters

_↩ _o	Offset in array of first element.
_↩ _l	Array of dimension lengths.
_↩ _s	Array of dimension strides between array elements.

References gslice().

gslice_array()

```
template<typename _Tp>
std::gslice_array< _Tp >::gslice_array (
            const gslice_array< _Tp > & __a)  [inline]
```

Copy constructor. Both slices refer to the same underlying array.
References gslice_array().
Referenced by gslice_array(), operator=(), and operator>>=().

indirect_array()

```
template<typename _Tp>
std::indirect_array< _Tp >::indirect_array (
            const indirect_array< _Tp > & __a)  [inline]
```

Copy constructor. Both slices refer to the same underlying array.
References indirect_array().
Referenced by indirect_array(), operator=(), and operator>>=().

mask_array()

```
template<typename _Tp>
std::mask_array< _Tp >::mask_array (
            const mask_array< _Tp > & __a)  [inline]
```

Copy constructor. Both slices refer to the same underlying array.
References mask_array().
Referenced by mask_array(), operator=(), and operator>>=().

slice() [1/2]

```
std::slice::slice ()  [inline]
```

Construct an empty slice.
Referenced by operator==.

slice() [2/2]

```
std::slice::slice (
            size_t __o,
            size_t __d,
            size_t __s)  [inline]
```

Construct a slice.

Parameters

_↩ _*o*	Offset in array of first element.
_↩ _*d*	Number of elements in slice.
_↩ _*s*	Stride between array elements.

slice_array()

```
template<typename _Tp>
std::slice_array< _Tp >::slice_array (
            const slice_array< _Tp > & __a)  [inline]
```

Copy constructor. Both slices refer to the same underlying array.
References slice_array().
Referenced by slice_array(), operator=(), and operator>>=().

valarray() [1/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray ()  [inline], [noexcept]
```

Construct an empty array.

valarray() [2/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
            const _Tp & __t,
            size_t __n)  [inline]
```

Construct an array with *n* elements initialized to *t*.

valarray() [3/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
            const gslice_array< _Tp > & __ga)  [inline]
```

Construct an array with the same size and values in *ga*.

valarray() [4/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
            const indirect_array< _Tp > & __ia)  [inline]
```

Construct an array with the same size and values in *ia*.

valarray() [5/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
            const mask_array< _Tp > & __ma)  [inline]
```

Construct an array with the same size and values in *ma*.

valarray() [6/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
            const slice_array< _Tp > & __sa)  [inline]
```

Construct an array with the same size and values in *sa*.

valarray() [7/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
            const valarray< _Tp > & __v)  [inline]
```

Copy constructor.

valarray() [8/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
            initializer_list< _Tp > __l)  [inline]
```

Construct an array with an initializer_list of values.

valarray() [9/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
            size_t __n)  [inline], [explicit]
```

Construct an array with *n* elements.

valarray() [10/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
            valarray< _Tp > && __v)  [inline], [noexcept]
```

Move constructor.

~gslice()

```
std::gslice::~gslice ()  [inline]
```

Destructor.

apply() [1/2]

```
template<class _Tp>
_Expr< _ValFunClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::apply (
            _Tp  __func(_Tp) const  [inline]
```

Apply a function to the array.
Returns a new valarray with elements assigned to the result of applying __func to the corresponding element of this array. The new array has the same size as this one.

Parameters

__func	Function of Tp returning Tp to apply.

Returns

New valarray with transformed elements.

apply() [2/2]

```
template<class _Tp>
_Expr< _RefFunClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::apply (
            _Tp  __funcconst _Tp &) const  [inline]
```

Apply a function to the array.
Returns a new valarray with elements assigned to the result of applying __func to the corresponding element of this array. The new array has the same size as this one.

Parameters

__func	Function of const Tp& returning Tp to apply.

Returns

New valarray with transformed elements.

begin() [1/2]

```
template<class _Tp>
const _Tp * std::begin (
            const valarray< _Tp > & __va)  [inline], [noexcept]
```

Return an iterator pointing to the first element of the const valarray.

Parameters

__va	valarray.

begin() [2/2]

```
template<class _Tp>
_Tp * std::begin (
            valarray< _Tp > & __va)  [inline], [noexcept]
```

Return an iterator pointing to the first element of the valarray.

Parameters

__va	valarray.

Referenced by cbegin(), std::vector< _Tp, _Alloc >::insert(), std::list< _Tp, _Alloc >::merge(), std::list< _Tp, _Alloc >::merge(), std::list< _Tp, _Alloc >::remove(), std::list< _Tp, _Alloc >::remove_if(), std::vector< _Tp, polymorphic_allocator< _Tp > >< _BigBlock std::list< _Tp, _Alloc >::sort(), std::list< _Tp, _Alloc >::sort(), std::list< _Tp, _Alloc >::unique(), and std::list< _Tp, _Alloc >::unique().

cshift()

```
template<class _Tp>
valarray< _Tp > std::valarray< _Tp >::cshift (
            int __n) const  [inline]
```

Return a rotated array.
A new valarray is constructed as a copy of this array with elements in shifted positions. For an element with index i, the new position is (i - n) % size(). The new valarray has the same size as the current one. Elements that are shifted beyond the array bounds are shifted into the other end of the array. No elements are lost.
Positive arguments shift toward index 0, wrapping around the top. Negative arguments shift towards the top, wrapping around to 0.

Parameters

_↩ _*n*	Number of element positions to rotate.

Returns

New valarray with elements in shifted positions.

end() [1/2]

```
template<class _Tp>
const _Tp * std::end (
            const valarray< _Tp > & __va)  [inline], [noexcept]
```

Return an iterator pointing to one past the last element of the const valarray.

Parameters

__va	valarray.

end() [2/2]

```
template<class _Tp>
_Tp * std::end (
            valarray< _Tp > & __va)  [inline], [noexcept]
```

Return an iterator pointing to one past the last element of the valarray.

Parameters

__va	valarray.

Referenced by cend(), std::vector< _Tp, _Alloc >::insert(), std::list< _Tp, _Alloc >::merge(), std::list< _Tp, _Alloc >::merge(), std::vector< _Tp, _Alloc >::operator=(), std::list< _Tp, _Alloc >::remove(), std::list< _Tp, _Alloc >::remove_if(), std::vector< _Tp, polymorphic_allocator< _Tp > >< _BigBlock >::reserve(), std::list< _Tp, _Alloc >::resize(), std::list< _Tp, _Alloc >::resize(), std::list< _Tp, _Alloc >::sort(), std::list< _Tp, _Alloc >::sort(), std::list< _Tp, _Alloc >::unique(), and std::list< _Tp, _Alloc >::unique().

max()

```
template<typename _Tp>
_Tp std::valarray< _Tp >::max () const  [inline]
```

Return the maximum element using operator<().

min()

```
template<typename _Tp>
_Tp std::valarray< _Tp >::min () const  [inline]
```

Return the minimum element using operator<().

operator=() [1/20]

```
gslice & std::gslice::operator= (
            const gslice & __g)  [inline]
```

Assignment operator.
References gslice().

operator=() [2/20]

```
template<typename _Tp>
void std::gslice_array< _Tp >::operator= (
            const _Tp & __t) const  [inline]
```

Assign all slice elements to *t*.

operator=() [3/20]

```
template<typename _Tp>
gslice_array< _Tp > & std::gslice_array< _Tp >::operator= (
            const gslice_array< _Tp > & __a)  [inline]
```

Assignment operator. Assigns slice elements to corresponding elements of *a*.
References gslice_array().

operator=() [4/20]

```
template<typename _Tp>
void std::gslice_array< _Tp >::operator= (
            const valarray< _Tp > & __v) const  [inline]
```

Assign slice elements to corresponding elements of *v*.
References std::valarray< _Tp >::size().

operator=() [5/20]

```
template<typename _Tp>
void std::indirect_array< _Tp >::operator= (
            const _Tp & __t) const  [inline]
```

Assign all slice elements to *t*.

operator=() [6/20]

```
template<typename _Tp>
indirect_array< _Tp > & std::indirect_array< _Tp >::operator= (
            const indirect_array< _Tp > & __a)  [inline]
```

Assignment operator. Assigns elements to corresponding elements of *a*.
References indirect_array().

operator=() [7/20]

```
template<typename _Tp>
void std::indirect_array< _Tp >::operator= (
            const valarray< _Tp > & __v) const  [inline]
```

Assign slice elements to corresponding elements of *v*.

operator=() [8/20]

```
template<typename _Tp>
void std::mask_array< _Tp >::operator= (
            const _Tp & __t) const  [inline]
```

Assign all slice elements to *t*.

operator=() [9/20]

```
template<typename _Tp>
mask_array< _Tp > & std::mask_array< _Tp >::operator= (
            const mask_array< _Tp > & __a)  [inline]
```

Assignment operator. Assigns elements to corresponding elements of *a*.
References mask_array().

operator=() [10/20]

```
template<typename _Tp>
void std::slice_array< _Tp >::operator= (
            const _Tp & __t) const  [inline]
```

Assign all slice elements to *t*.

operator=() [11/20]

```
template<typename _Tp>
slice_array< _Tp > & std::slice_array< _Tp >::operator= (
            const slice_array< _Tp > & __a)  [inline]
```

Assignment operator. Assigns slice elements to corresponding elements of *a*.
References slice_array().

operator=() [12/20]

```
template<typename _Tp>
void std::slice_array< _Tp >::operator= (
            const valarray< _Tp > & __v) const  [inline]
```

Assign slice elements to corresponding elements of *v*.

operator=() [13/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
            const _Tp & __t)  [inline]
```

Assign elements to a value.
Assign all elements of array to *t*.

Parameters

↩ _↩ ↩ _↩ *t*	Value for elements.

operator=() [14/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
            const gslice_array< _Tp > & __ga)  [inline]
```

Assign elements to an array subset.
Assign elements of array to values in *ga*. Results are undefined if *ga* does not have the same size as this array.

Parameters

__ga	Array slice to get values from.

operator=() [15/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
            const indirect_array< _Tp > & __ia)  [inline]
```

Assign elements to an array subset.
Assign elements of array to values in *ia*. Results are undefined if *ia* does not have the same size as this array.

Parameters

_↩ _ia	Array slice to get values from.

operator=() [16/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
            const mask_array< _Tp > & __ma)  [inline]
```

Assign elements to an array subset.
Assign elements of array to values in *ma*. Results are undefined if *ma* does not have the same size as this array.

Parameters

__ma	Array slice to get values from.

operator=() [17/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
            const slice_array< _Tp > & __sa)  [inline]
```

Assign elements to an array subset.
Assign elements of array to values in *sa*. Results are undefined if *sa* does not have the same size as this array.

Parameters

__sa	Array slice to get values from.

operator=() [18/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
            const valarray< _Tp > & __v)  [inline]
```

Assign elements to an array.
Assign elements of array to values in *v*.

Parameters

_↩ _v	Valarray to get values from.

operator=() [19/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
            initializer_list< _Tp > __l)  [inline]
```

Assign elements to an initializer_list.
Assign elements of array to values in *__l*. Results are undefined if *__l* does not have the same size as this array.

Parameters

↩ _↩ ↩ _↩ *l*	initializer_list to get values from.

operator=() [20/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
            valarray< _Tp > && __v)  [inline], [noexcept]
```

Move assign elements to an array.
Move assign elements of array to values in *v*.

Parameters

_↩ _v	Valarray to get values from.

operator[]() [1/9]

```
template<typename _Tp>
gslice_array< _Tp > std::valarray< _Tp >::operator[] (
            const gslice & __s)  [inline]
```

Return a reference to an array subset.
Returns a new valarray containing the elements of the array indicated by the gslice argument. The new valarray has the same size as the input gslice.

See also

gslice.

Parameters

_↩ _s	The source gslice.

Returns

New valarray containing elements in __s.

operator[]() [2/9]

```
template<typename _Tp>
_Expr< _GClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::operator[] (
            const gslice & __s) const  [inline]
```

Return an array subset.
Returns a slice_array referencing the elements of the array indicated by the slice argument.

See also

gslice.

Parameters

_↩ _s	The source slice.

Returns

Slice_array referencing elements indicated by __s.

operator[]() [3/9]

```
template<typename _Tp>
mask_array< _Tp > std::valarray< _Tp >::operator[] (
            const valarray< bool > & __m)  [inline]
```

Return a reference to an array subset.
Returns a new mask_array referencing the elements of the array indicated by the argument. The input is a valarray of bool which represents a bitmask indicating which elements are part of the subset. Elements of the array are part of the subset if the corresponding element of the argument is true.

Parameters

_↩ _m	The valarray bitmask.

Returns

New valarray containing elements indicated by __m.

operator[]() [4/9]

```
template<typename _Tp>
valarray< _Tp > std::valarray< _Tp >::operator[] (
            const valarray< bool > & __m) const  [inline]
```

Return an array subset.
Returns a new valarray containing the elements of the array indicated by the argument. The input is a valarray of bool which represents a bitmask indicating which elements should be copied into the new valarray. Each element of the array is added to the return valarray if the corresponding element of the argument is true.

Parameters

_↩ _*m*	The valarray bitmask.

Returns

New valarray containing elements indicated by __*m*.

operator[]() [5/9]

```
template<typename _Tp>
indirect_array< _Tp > std::valarray< _Tp >::operator[] (
            const valarray< size_t > & __i)  [inline]
```

Return a reference to an array subset.
Returns an indirect_array referencing the elements of the array indicated by the argument. The elements in the argument are interpreted as the indices of elements of this valarray to include in the subset. The returned indirect_array refers to these elements.

Parameters

↩ _↩ ↩ _↩ *i*	The valarray element index list.

Returns

Indirect_array referencing elements in __*i*.

operator[]() [6/9]

```
template<typename _Tp>
_Expr< _IClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::operator[] (
            const valarray< size_t > & __i) const  [inline]
```

Return an array subset.
Returns a new valarray containing the elements of the array indicated by the argument. The elements in the argument are interpreted as the indices of elements of this valarray to copy to the return valarray.

Parameters

↩ _↩ ↩ _↩ *i*	The valarray element index list.

Returns

New valarray containing elements in __*s*.

operator[]() [7/9]

```
template<typename _Tp>
_Tp & std::valarray< _Tp >::operator[] (
            size_t __i)  [inline], [noexcept]
```

Return a reference to the i'th array element.

Parameters

↩ _↩ ↩ _↩ *i*	Index of element to return.

Returns

Reference to the i'th element.

operator[]() [8/9]

```
template<typename _Tp>
slice_array< _Tp > std::valarray< _Tp >::operator[] (
            slice __s)  [inline]
```

Return a reference to an array subset.

Returns a new valarray containing the elements of the array indicated by the slice argument. The new valarray has the same size as the input slice.

See also

slice.

Parameters

_↩ *_s*	The source slice.

Returns

New valarray containing elements in __*s*.

operator[]() [9/9]

```
template<typename _Tp>
_Expr< _SClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::operator[] (
            slice __s) const  [inline]
```

Return an array subset.
Returns a new valarray containing the elements of the array indicated by the slice argument. The new valarray has the same size as the input slice.

See also

slice.

Parameters

_↩ _s	The source slice.

Returns

New valarray containing elements in __s.

resize()

```
template<class _Tp>
void std::valarray< _Tp >::resize (
            size_t __size,
            _Tp __c = _Tp())  [inline]
```

Resize array.
Resize this array to *size* and set all elements to *c*. All references and iterators are invalidated.

Parameters

__size	New array size.
__c	New value for all elements.

shift()

```
template<class _Tp>
valarray< _Tp > std::valarray< _Tp >::shift (
            int __n) const  [inline]
```

Return a shifted array.
A new valarray is constructed as a copy of this array with elements in shifted positions. For an element with index i, the new position is i - n. The new valarray has the same size as the current one. New elements without a value are set to 0. Elements whose new position is outside the bounds of the array are discarded.
Positive arguments shift toward index 0, discarding elements [0, n). Negative arguments discard elements from the top of the array.

Parameters

_↩ _n	Number of element positions to shift.

Returns

New valarray with elements in shifted positions.

size() [1/3]

```
valarray< size_t > std::gslice::size () const  [inline]
```
Return array of sizes of slice dimensions.

size() [2/3]

```
size_t std::slice::size () const  [inline]
```
Return size of slice.

size() [3/3]

```
template<class _Tp>
size_t std::valarray< _Tp >::size () const  [inline]
```
Return the number of elements in array.
Referenced by std::gslice_array< _Tp >::operator=().

start() [1/2]

```
size_t std::gslice::start () const  [inline]
```
Return array offset of first slice element.

start() [2/2]

```
size_t std::slice::start () const  [inline]
```
Return array offset of first slice element.

stride() [1/2]

```
valarray< size_t > std::gslice::stride () const  [inline]
```
Return array of array strides for each dimension.

stride() [2/2]

```
size_t std::slice::stride () const  [inline]
```
Return array stride of slice.

sum()

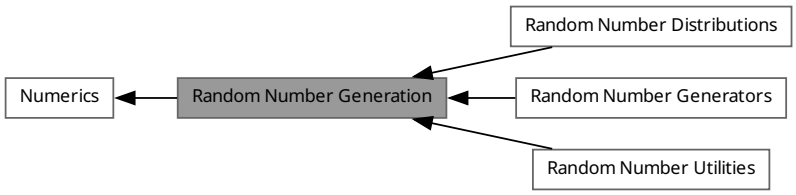
```
template<class _Tp>
_Tp std::valarray< _Tp >::sum () const  [inline]
```
Return the sum of all elements in the array.
Accumulates the sum of all elements into a Tp using +=. The order of adding the elements is unspecified.

swap()

```
template<class _Tp>
void std::valarray< _Tp >::swap (
            valarray< _Tp > & __v)  [inline], [noexcept]
```
Swap.

3.11.7 Random Number Generation

Collaboration diagram for Random Number Generation:



Topics

- Random Number Distributions
- Random Number Generators
- Random Number Utilities

Functions

- template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator>
  _RealType std::generate_canonical (_UniformRandomNumberGenerator &__g)

3.11.7.1 Detailed Description

A facility for generating random numbers on selected distributions.

3.11.7.2 Function Documentation

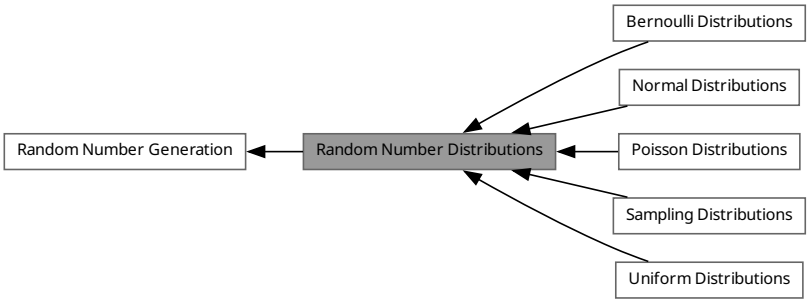
generate_canonical()

```
template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator>
_RealType std::generate_canonical (
            _UniformRandomNumberGenerator & __g)
```

A function template for converting the output of a (integral) uniform random number generator to a floatng point result in the range [0-1).
References std::__numeric_limits_base::digits, std::numeric_limits< _Tp >::epsilon(), log(), max(), and min().

3.11.7.3 Random Number Distributions

Collaboration diagram for Random Number Distributions:



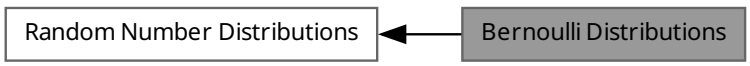
Topics

- Bernoulli Distributions
- Normal Distributions
- Poisson Distributions
- Sampling Distributions
- Uniform Distributions

3.11.7.3.1 Detailed Description

3.11.7.3.2 Bernoulli Distributions

Collaboration diagram for Bernoulli Distributions:



Classes

- class std::bernoulli_distribution
- class std::binomial_distribution< _IntType >
- class std::geometric_distribution< _IntType >
- class std::negative_binomial_distribution< _IntType >

Functions

- template<typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::bernoulli_distribution &__x)

- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::geometric_distribution< _IntType > &__x)
- template<typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::bernoulli_distribution &__x)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::geometric_distribution< _IntType > &__x)

Detailed Description

Function Documentation

operator<<() [1/2]

```
template<typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & std::operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::bernoulli_distribution & __x)
```

Inserts a bernoulli_distribution random number distribution __x into the output stream __os.

Parameters

__os	An output stream.
__x	A bernoulli_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

operator<<() [2/2]

```
template<typename _IntType, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & std::operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::geometric_distribution< _IntType > & __x)
```

Inserts a geometric_distribution random number distribution __x into the output stream __os.

Parameters

__os	An output stream.
__x	A geometric_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

References std::numeric_limits< _Tp >::epsilon(), log(), and std::numeric_limits< _Tp >::max().

operator>>() [1/2]

```
template<typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & std::operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            std::bernoulli_distribution & __x)  [inline]
```

Extracts a bernoulli_distribution random number distribution __x from the input stream __is.

Parameters

_↩ _is	An input stream.
_↩ _x	A bernoulli_distribution random number generator engine.

Returns

The input stream with __x extracted or in an error state.

References std::bernoulli_distribution::param().

operator>>() [2/2]

```
template<typename _IntType, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & std::operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            std::geometric_distribution< _IntType > & __x)
```

Extracts a geometric_distribution random number distribution __x from the input stream __is.

Parameters

_↩ _is	An input stream.
_↩ _x	A geometric_distribution random number generator engine.

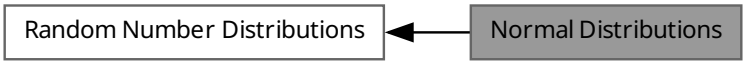
Returns

The input stream with __x extracted or in an error state.

References std::ios_base::flags(), and std::geometric_distribution< _IntType >::param().

3.11.7.3.3 Normal Distributions

Collaboration diagram for Normal Distributions:



Classes

- class std::cauchy_distribution< _RealType >
- class std::chi_squared_distribution< _RealType >
- class std::fisher_f_distribution< _RealType >
- class std::lognormal_distribution< _RealType >
- class std::normal_distribution< _RealType >
- class std::student_t_distribution< _RealType >

Functions

- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::cauchy_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::cauchy_distribution< _RealType > &__x)

Detailed Description

Function Documentation

operator<<()

```
template<typename _RealType, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & std::operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::cauchy_distribution< _RealType > & __x)
```

Inserts a cauchy_distribution random number distribution __x into the output stream __os.

Parameters

__os	An output stream.
__x	A cauchy_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

References tan().

operator>>()

```
template<typename _RealType, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & std::operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            std::cauchy_distribution< _RealType > & __x)
```

Extracts a cauchy_distribution random number distribution __x from the input stream __is.

Parameters

_↩ _is	An input stream.
_↩ _x	A cauchy_distribution random number generator engine.

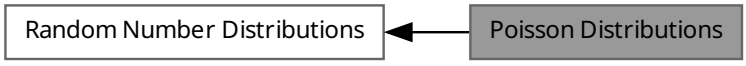
Returns

The input stream with __x extracted or in an error state.

References std::ios_base::flags(), and std::cauchy_distribution< _RealType >::param().

3.11.7.3.4 Poisson Distributions

Collaboration diagram for Poisson Distributions:



Classes

- class std::exponential_distribution< _RealType >
- class std::extreme_value_distribution< _RealType >
- class std::gamma_distribution< _RealType >
- class std::poisson_distribution< _IntType >
- class std::weibull_distribution< _RealType >

Functions

- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::exponential_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::extreme_value_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::weibull_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::exponential_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::extreme_value_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::weibull_distribution< _RealType > &__x)

Detailed Description

Function Documentation

operator<<() [1/3]

```
template<typename _RealType, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & std::operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::exponential_distribution< _RealType > & __x)
```

Inserts a exponential_distribution random number distribution __x into the output stream __os.

Parameters

__os	An output stream.
__x	A exponential_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

References log().

operator<<() [2/3]

```
template<typename _RealType, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & std::operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::extreme_value_distribution< _RealType > & __x)
```

Inserts a extreme_value_distribution random number distribution __x into the output stream __os.

Parameters

__os	An output stream.
__x	A extreme_value_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

References log().

operator<<() [3/3]

```
template<typename _RealType, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & std::operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::weibull_distribution< _RealType > & __x)
```

Inserts a weibull_distribution random number distribution __x into the output stream __os.

Parameters

__os	An output stream.
__x	A weibull_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

References log(), and pow().

operator>>() [1/3]

```
template<typename _RealType, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & std::operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            std::exponential_distribution< _RealType > & __x)
```

Extracts a exponential_distribution random number distribution __x from the input stream __is.

Parameters

_↩ _is	An input stream.
_↩ _x	A exponential_distribution random number generator engine.

Returns

The input stream with __x extracted or in an error state.

References std::ios_base::flags(), and std::exponential_distribution< _RealType >::param().

operator>>() [2/3]

```
template<typename _RealType, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & std::operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            std::extreme_value_distribution< _RealType > & __x)
```

Extracts a extreme_value_distribution random number distribution __x from the input stream __is.

Parameters

_↩ _is	An input stream.
_↩ _x	A extreme_value_distribution random number generator engine.

Returns

The input stream with __x extracted or in an error state.

References std::ios_base::flags(), and std::extreme_value_distribution< _RealType >::param().

operator>>() [3/3]

```
template<typename _RealType, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & std::operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            std::weibull_distribution< _RealType > & __x)
```

Extracts a weibull_distribution random number distribution __x from the input stream __is.

Parameters

_↩ _is	An input stream.
_↩ _x	A weibull_distribution random number generator engine.

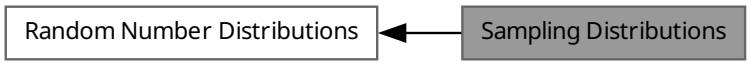
Returns

The input stream with __x extracted or in an error state.

References std::ios_base::flags(), and std::weibull_distribution< _RealType >::param().

3.11.7.3.5 Sampling Distributions

Collaboration diagram for Sampling Distributions:



Classes

- class std::discrete_distribution< _IntType >
- class std::piecewise_constant_distribution< _RealType >
- class std::piecewise_linear_distribution< _RealType >

Detailed Description

3.11.7.3.6 Uniform Distributions

Collaboration diagram for Uniform Distributions:

Classes

- class std::uniform_int_distribution< _IntType >
- class std::uniform_real_distribution< _RealType >

Functions

- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_int_distribution< _IntType > &)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_real_distribution< _RealType > &)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_int_distribution< _IntType > &)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_real_distribution< _RealType > &)

Detailed Description

Function Documentation

operator<<() [1/2]

```
template<typename _IntType, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & std::operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::uniform_int_distribution< _IntType > & __x)
```

Inserts a uniform_int_distribution random number distribution __x into the output stream os.

Parameters

__os	An output stream.
__x	A uniform_int_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

References std::ios_base::flags().

operator<<() [2/2]

```
template<typename _RealType, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & std::operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::uniform_real_distribution< _RealType > & __x)
```

Inserts a uniform_real_distribution random number distribution __x into the output stream __os.

Parameters

__os	An output stream.
__x	A uniform_real_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

operator>>() [1/2]

```
template<typename _IntType, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & std::operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            std::uniform_int_distribution< _IntType > & __x)
```

Extracts a uniform_int_distribution random number distribution __x from the input stream __is.

Parameters

_↩ _is	An input stream.
_↩ _x	A uniform_int_distribution random number generator engine.

Returns

The input stream with __x extracted or in an error state.

References std::ios_base::flags(), and std::uniform_int_distribution< _IntType >::param().

operator>>() [2/2]

```
template<typename _RealType, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & std::operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            std::uniform_real_distribution< _RealType > & __x)
```

Extracts a uniform_real_distribution random number distribution __x from the input stream __is.

Parameters

_↩ _is	An input stream.
_↩ _x	A uniform_real_distribution random number generator engine.

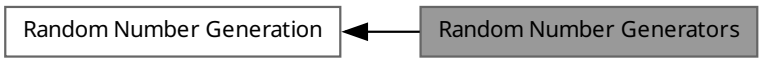
Returns

The input stream with __x extracted or in an error state.

References std::ios_base::flags(), and std::uniform_real_distribution< _RealType >::param().

3.11.7.4 Random Number Generators

Collaboration diagram for Random Number Generators:



Classes

- class std::discard_block_engine< _RandomNumberEngine, __p, __r >
- class std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >
- class std::linear_congruential_engine< _UIntType, __a, __c, __m >
- class std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >
- class std::random_device
- class std::shuffle_order_engine< _RandomNumberEngine, __k >
- class std::subtract_with_carry_engine< _UIntType, __w, __s, __r >

Typedefs

- typedef minstd_rand0 **std::default_random_engine**
- typedef shuffle_order_engine< minstd_rand0, 256 > **std::knuth_b**
- typedef linear_congruential_engine< uint_fast32_t, 48271UL, 0UL, 2147483647UL > std::minstd_rand

- typedef linear_congruential_engine< uint_fast32_t, 16807UL, 0UL, 2147483647UL > std::minstd_rand0
- typedef mersenne_twister_engine< uint_fast32_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL > std::mt19937
- typedef mersenne_twister_engine< uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67fffeda60000ULL, 37, 0xfff7eee000000000ULL, 43, 6364136223846793005ULL > std::mt19937_64
- typedef discard_block_engine< ranlux24_base, 223, 23 > **std::ranlux24**
- typedef subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 > **std::ranlux24_base**
- typedef discard_block_engine< ranlux48_base, 389, 11 > **std::ranlux48**
- typedef subtract_with_carry_engine< uint_fast64_t, 48, 5, 12 > **std::ranlux48_base**

Functions

- template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__x)

3.11.7.4.1 Detailed Description

These classes define objects which provide random or pseudorandom numbers, either from a discrete or a continuous interval. The random number generator supplied as a part of this library are all uniform random number generators which provide a sequence of random number uniformly distributed over their range.
A number generator is a function object with an operator() that takes zero arguments and returns a number.
A compliant random number generator must satisfy the following requirements.

Table 291 Random Number Generator Requirements

To be documented.

3.11.7.4.2 Typedef Documentation

minstd_rand

```
typedef linear_congruential_engine<uint_fast32_t, 48271UL, 0UL, 2147483647UL> std::minstd_rand
```

An alternative LCR (Lehmer Generator function).

minstd_rand0

```
typedef linear_congruential_engine<uint_fast32_t, 16807UL, 0UL, 2147483647UL> std::minstd_rand0
```

The classic Minimum Standard rand0 of Lewis, Goodman, and Miller.

mt19937

```
typedef mersenne_twister_engine< uint_fast32_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL,
7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL> std::mt19937
```

The classic Mersenne Twister.
Reference: M. Matsumoto and T. Nishimura, Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator, ACM Transactions on Modeling and Computer Simulation, Vol. 8, No. 1, January 1998, pp 3-30.

mt19937_64

```
typedef mersenne_twister_engine< uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29,
0x5555555555555555ULL, 17, 0x71d67fffeda60000ULL, 37, 0xfff7eee000000000ULL, 43, 6364136223846793005ULL>
std::mt19937_64
```

An alternative Mersenne Twister.

3.11.7.4.3 Function Documentation

operator<<()

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename
_Traits>
std::basic_ostream< _CharT, _Traits > & std::operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > & __x)
```

Inserts the current state of a independent_bits_engine random number generator engine `__x` into the output stream `__os`.

Parameters

__os	An output stream.
__x	A independent_bits_engine random number generator engine.

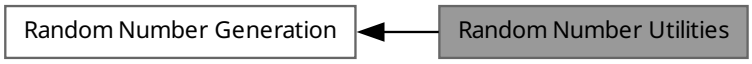
Returns

The output stream with the state of `__x` inserted or in an error state.

References std::__numeric_limits_base::digits.

3.11.7.5 Random Number Utilities

Collaboration diagram for Random Number Utilities:



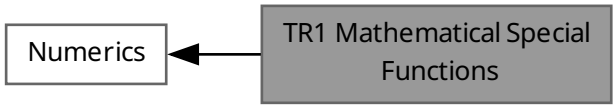
Classes

- class std::seed_seq

3.11.7.5.1 Detailed Description

3.11.8 TR1 Mathematical Special Functions

Collaboration diagram for TR1 Mathematical Special Functions:



Functions

- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::assoc_laguerre (unsigned int __n, unsigned int __m, _Tp __x)
- float **std::tr1::assoc_laguerref** (unsigned int __n, unsigned int __m, float __x)
- long double **std::tr1::assoc_laguerrel** (unsigned int __n, unsigned int __m, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::assoc_legendre (unsigned int __l, unsigned int __m, _Tp __x)
- float **std::tr1::assoc_legendref** (unsigned int __l, unsigned int __m, float __x)
- long double **std::tr1::assoc_legendrel** (unsigned int __l, unsigned int __m, long double __x)
- template<typename _Tpx, typename _Tpy>
  __gnu_cxx::__promote_2< _Tpx, _Tpy >::__type std::tr1::beta (_Tpx __x, _Tpy __y)
- float **std::tr1::betaf** (float __x, float __y)
- long double **std::tr1::betal** (long double __x, long double __y)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::comp_ellint_1 (_Tp __k)
- float **std::tr1::comp_ellint_1f** (float __k)
- long double **std::tr1::comp_ellint_1l** (long double __k)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::comp_ellint_2 (_Tp __k)
- float **std::tr1::comp_ellint_2f** (float __k)
- long double **std::tr1::comp_ellint_2l** (long double __k)
- template<typename _Tp, typename _Tpn>
  __gnu_cxx::__promote_2< _Tp, _Tpn >::__type std::tr1::comp_ellint_3 (_Tp __k, _Tpn __nu)
- float **std::tr1::comp_ellint_3f** (float __k, float __nu)
- long double **std::tr1::comp_ellint_3l** (long double __k, long double __nu)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_i (_Tpnu __nu, _Tp __x)
- float **std::tr1::cyl_bessel_if** (float __nu, float __x)
- long double **std::tr1::cyl_bessel_il** (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_j (_Tpnu __nu, _Tp __x)
- float **std::tr1::cyl_bessel_jf** (float __nu, float __x)
- long double **std::tr1::cyl_bessel_jl** (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_k (_Tpnu __nu, _Tp __x)
- float **std::tr1::cyl_bessel_kf** (float __nu, float __x)
- long double **std::tr1::cyl_bessel_kl** (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_neumann (_Tpnu __nu, _Tp __x)
- float **std::tr1::cyl_neumannf** (float __nu, float __x)
- long double **std::tr1::cyl_neumannl** (long double __nu, long double __x)
- template<typename _Tp, typename _Tpp>
  __gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::tr1::ellint_1 (_Tp __k, _Tpp __phi)
- float **std::tr1::ellint_1f** (float __k, float __phi)
- long double **std::tr1::ellint_1l** (long double __k, long double __phi)
- template<typename _Tp, typename _Tpp>
  __gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::tr1::ellint_2 (_Tp __k, _Tpp __phi)
- float **std::tr1::ellint_2f** (float __k, float __phi)
- long double **std::tr1::ellint_2l** (long double __k, long double __phi)
- template<typename _Tp, typename _Tpn, typename _Tpp>
  __gnu_cxx::__promote_3< _Tp, _Tpn, _Tpp >::__type std::tr1::ellint_3 (_Tp __k, _Tpn __nu, _Tpp __phi)

- float **std::tr1::ellint_3f** (float __k, float __nu, float __phi)
- long double **std::tr1::ellint_3l** (long double __k, long double __nu, long double __phi)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::expint (_Tp __x)
- float **std::tr1::expintf** (float __x)
- long double **std::tr1::expintl** (long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::hermite (unsigned int __n, _Tp __x)
- float **std::tr1::hermitef** (unsigned int __n, float __x)
- long double **std::tr1::hermitel** (unsigned int __n, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::laguerre (unsigned int __n, _Tp __x)
- float **std::tr1::laguerref** (unsigned int __n, float __x)
- long double **std::tr1::laguerrel** (unsigned int __n, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::legendre (unsigned int __n, _Tp __x)
- float **std::tr1::legendref** (unsigned int __n, float __x)
- long double **std::tr1::legendrel** (unsigned int __n, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::riemann_zeta (_Tp __x)
- float **std::tr1::riemann_zetaf** (float __x)
- long double **std::tr1::riemann_zetal** (long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::sph_bessel (unsigned int __n, _Tp __x)
- float **std::tr1::sph_besself** (unsigned int __n, float __x)
- long double **std::tr1::sph_bessell** (unsigned int __n, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::sph_legendre (unsigned int __l, unsigned int __m, _Tp __theta)
- float **std::tr1::sph_legendref** (unsigned int __l, unsigned int __m, float __theta)
- long double **std::tr1::sph_legendrel** (unsigned int __l, unsigned int __m, long double __theta)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::sph_neumann (unsigned int __n, _Tp __x)
- float **std::tr1::sph_neumannf** (unsigned int __n, float __x)
- long double **std::tr1::sph_neumannl** (unsigned int __n, long double __x)

3.11.8.1 Detailed Description

A collection of advanced mathematical special functions.

3.11.8.2 Function Documentation

assoc_laguerre()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::tr1::assoc_laguerre (
            unsigned int __n,
            unsigned int __m,
            _Tp __x)  [inline]
```

5.2.1.1 Associated Laguerre polynomials.

assoc_legendre()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::tr1::assoc_legendre (
             unsigned int __l,
             unsigned int __m,
             _Tp __x)  [inline]
```

5.2.1.2 Associated Legendre functions.

beta()

```
template<typename _Tpx, typename _Tpy>
__gnu_cxx::__promote_2< _Tpx, _Tpy >::__type std::tr1::beta (
             _Tpx __x,
             _Tpy __y)  [inline]
```

5.2.1.3 Beta functions.

comp_ellint_1()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::tr1::comp_ellint_1 (
             _Tp __k)  [inline]
```

5.2.1.4 Complete elliptic integrals of the first kind.

comp_ellint_2()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::tr1::comp_ellint_2 (
             _Tp __k)  [inline]
```

5.2.1.5 Complete elliptic integrals of the second kind.

comp_ellint_3()

```
template<typename _Tp, typename _Tpn>
__gnu_cxx::__promote_2< _Tp, _Tpn >::__type std::tr1::comp_ellint_3 (
             _Tp __k,
             _Tpn __nu)  [inline]
```

5.2.1.6 Complete elliptic integrals of the third kind.

cyl_bessel_i()

```
template<typename _Tpnu, typename _Tp>
__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_i (
             _Tpnu __nu,
             _Tp __x)  [inline]
```

5.2.1.8 Regular modified cylindrical Bessel functions.

cyl_bessel_j()

```
template<typename _Tpnu, typename _Tp>
__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_j (
             _Tpnu __nu,
             _Tp __x)  [inline]
```

5.2.1.9 Cylindrical Bessel functions (of the first kind).

cyl_bessel_k()

```
template<typename _Tpnu, typename _Tp>
__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_k (
            _Tpnu __nu,
            _Tp __x)  [inline]
```

5.2.1.10 Irregular modified cylindrical Bessel functions.

cyl_neumann()

```
template<typename _Tpnu, typename _Tp>
__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_neumann (
            _Tpnu __nu,
            _Tp __x)  [inline]
```

5.2.1.11 Cylindrical Neumann functions.

ellint_1()

```
template<typename _Tp, typename _Tpp>
__gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::tr1::ellint_1 (
            _Tp __k,
            _Tpp __phi)  [inline]
```

5.2.1.12 Incomplete elliptic integrals of the first kind.

ellint_2()

```
template<typename _Tp, typename _Tpp>
__gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::tr1::ellint_2 (
            _Tp __k,
            _Tpp __phi)  [inline]
```

5.2.1.13 Incomplete elliptic integrals of the second kind.

ellint_3()

```
template<typename _Tp, typename _Tpn, typename _Tpp>
__gnu_cxx::__promote_3< _Tp, _Tpn, _Tpp >::__type std::tr1::ellint_3 (
            _Tp __k,
            _Tpn __nu,
            _Tpp __phi)  [inline]
```

5.2.1.14 Incomplete elliptic integrals of the third kind.

expint()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::tr1::expint (
            _Tp __x)  [inline]
```

5.2.1.15 Exponential integrals.

hermite()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::tr1::hermite (
            unsigned int __n,
            _Tp __x)  [inline]
```

5.2.1.16 Hermite polynomials.

laguerre()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::tr1::laguerre (
             unsigned int __n,
             _Tp __x)  [inline]
```

5.2.1.18 Laguerre polynomials.

legendre()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::tr1::legendre (
             unsigned int __n,
             _Tp __x)  [inline]
```

5.2.1.19 Legendre polynomials.

riemann_zeta()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::tr1::riemann_zeta (
             _Tp __x)  [inline]
```

5.2.1.20 Riemann zeta function.

sph_bessel()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::tr1::sph_bessel (
             unsigned int __n,
             _Tp __x)  [inline]
```

5.2.1.21 Spherical Bessel functions.

sph_legendre()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::tr1::sph_legendre (
             unsigned int __l,
             unsigned int __m,
             _Tp __theta)  [inline]
```

5.2.1.22 Spherical associated Legendre functions.

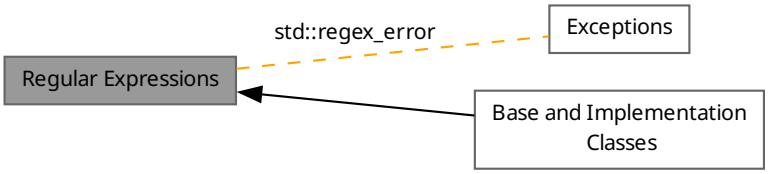
sph_neumann()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::tr1::sph_neumann (
             unsigned int __n,
             _Tp __x)  [inline]
```

5.2.1.23 Spherical Neumann functions.

3.12 Regular Expressions

Collaboration diagram for Regular Expressions:



Topics

- Base and Implementation Classes

Namespaces

- namespace std::regex_constants

Classes

- class std::basic_regex< _Ch_type, _Rx_traits >
- class std::match_results< _Bi_iter, _Alloc >
- class std::regex_error
- class std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >
- class std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >
- class std::regex_traits< _Ch_type >
- class std::sub_match< _BiIter >

Typedefs

- typedef match_results< const char * > **std::cmatch**
- typedef regex_iterator< const char * > **std::cregex_iterator**
- typedef regex_token_iterator< const char * > std::cregex_token_iterator
- typedef sub_match< const char * > std::csub_match
- typedef basic_regex< char > std::regex
- typedef match_results< string::const_iterator > **std::smatch**
- typedef regex_iterator< string::const_iterator > **std::sregex_iterator**
- typedef regex_token_iterator< string::const_iterator > std::sregex_token_iterator
- typedef sub_match< string::const_iterator > std::ssub_match
- typedef match_results< const wchar_t * > **std::wcmatch**
- typedef regex_iterator< const wchar_t * > **std::wcregex_iterator**
- typedef regex_token_iterator< const wchar_t * > std::wcregex_token_iterator
- typedef sub_match< const wchar_t * > std::wcsub_match
- typedef basic_regex< wchar_t > std::wregex
- typedef match_results< wstring::const_iterator > **std::wsmatch**
- typedef regex_iterator< wstring::const_iterator > **std::wsregex_iterator**
- typedef regex_token_iterator< wstring::const_iterator > std::wsregex_token_iterator
- typedef sub_match< wstring::const_iterator > std::wssub_match

Functions

- template<typename _ForwardIterator>
  std::basic_regex (_ForwardIterator, _ForwardIterator, regex_constants::syntax_option_type={}) -> basic_↩ regex< typename iterator_traits< _ForwardIterator >::value_type >
- template<typename _Bi_iter, typename _Alloc>
  bool operator== (const match_results< _Bi_iter, _Alloc > &__m1, const match_results< _Bi_iter, _Alloc > &_↩ _m2)
- template<typename _Ch_type, typename _Rx_traits>
  void swap (basic_regex< _Ch_type, _Rx_traits > &__lhs, basic_regex< _Ch_type, _Rx_traits > &__rhs) noex- cept
- template<typename _Bi_iter, typename _Alloc>
  void swap (match_results< _Bi_iter, _Alloc > &__lhs, match_results< _Bi_iter, _Alloc > &__rhs) noexcept

Matching, Searching, and Replacing

- template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits>
  bool std::regex_match (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>
  bool std::regex_match (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Ch_type, typename _Alloc, typename _Rx_traits>
  bool std::regex_match (const _Ch_type *__s, match_results< const _Ch_type *, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_constants::match_default)
- template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>
  bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>
  bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &&, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &, const basic_regex< _Ch_type, _Rx_traits > &, regex_constants::match_flag_type=regex_constants::match_default)=delete
- template<typename _Ch_type, class _Rx_traits>
  bool std::regex_match (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_constants::match_default)
- template<typename _Ch_traits, typename _Str_allocator, typename _Ch_type, typename _Rx_traits>
  bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Str_allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits>
  bool std::regex_search (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>
  bool std::regex_search (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Ch_type, class _Alloc, class _Rx_traits>
  bool std::regex_search (const _Ch_type *__s, match_results< const _Ch_type *, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)
- template<typename _Ch_type, typename _Rx_traits>
  bool std::regex_search (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits > &__↩ e, regex_constants::match_flag_type __f=regex_constants::match_default)
- template<typename _Ch_traits, typename _String_allocator, typename _Ch_type, typename _Rx_traits>
  bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _String_allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __flags=regex_constants::match_default)

- template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>
  bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)
- template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>
  bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &&, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &, const basic_regex< _Ch_type, _Rx_traits > &, regex_constants::match_flag_type=regex_constants::match_default)=delete
- template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa>
  _Out_iter std::regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__e, const basic_string< _Ch_type, _St, _Sa > &__fmt, regex_constants::match_flag_type __↩
  flags=regex_constants::match_default)
- template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type>
  _Out_iter std::regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _↩
  Rx_traits > &__e, const _Ch_type *__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa, typename _Fst, typename _Fsa>
  basic_string< _Ch_type, _St, _Sa > std::regex_replace (const basic_string< _Ch_type, _St, _Sa > &__↩
  s, const basic_regex< _Ch_type, _Rx_traits > &__e, const basic_string< _Ch_type, _Fst, _Fsa > &__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa>
  basic_string< _Ch_type, _St, _Sa > std::regex_replace (const basic_string< _Ch_type, _St, _Sa > &__s, const basic_regex< _Ch_type, _Rx_traits > &__e, const _Ch_type *__fmt, regex_constants::match_flag_type __↩
  flags=regex_constants::match_default)
- template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa>
  basic_string< _Ch_type > std::regex_replace (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx↩
  _traits > &__e, const basic_string< _Ch_type, _St, _Sa > &__fmt, regex_constants::match_flag_type __↩
  flags=regex_constants::match_default)
- template<typename _Rx_traits, typename _Ch_type>
  basic_string< _Ch_type > std::regex_replace (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits > &__e, const _Ch_type *__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)

- template<typename _BiIter>
  bool operator== (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__rhs)
- template<typename _BiIter>
  auto operator<=> (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__rhs) noexcept(__↩
  detail::__is_contiguous_iter< _BiIter >::value)
- template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc>
  bool operator== (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch↩
  _alloc > &__rhs)
- template<typename _Bi_iter, typename _Ch_traits, typename _Alloc>
  auto operator<=> (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _↩
  Alloc > &__rhs) noexcept(__detail::__is_contiguous_iter< _Bi_iter >::value)
- template<typename _Bi_iter>
  bool operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)
- template<typename _Bi_iter>
  auto operator<=> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs) noexcept(__detail::__is_contiguous_iter< _Bi_iter >::value)
- template<typename _Bi_iter>
  bool operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)
- template<typename _Bi_iter>
  auto operator<=> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs) noexcept(__detail::__is_contiguous_iter< _Bi_iter >::value)

- template<typename _Ch_type, typename _Ch_traits, typename _Bi_iter>
  basic_ostream< _Ch_type, _Ch_traits > & operator<< (basic_ostream< _Ch_type, _Ch_traits > &__os, const sub_match< _Bi_iter > &__m)

3.12.1 Detailed Description

A facility for performing regular expression pattern matching.

Since

C++11

3.12.2 Typedef Documentation

cregex_token_iterator

`typedef regex_token_iterator<const char*> std::cregex_token_iterator`
Token iterator for C-style NULL-terminated strings.

csub_match

`typedef sub_match<const char*> std::csub_match`
Standard regex submatch over a C-style null-terminated string.

regex

`typedef basic_regex<char> std::regex`
Standard regular expressions.

sregex_token_iterator

`typedef regex_token_iterator<string::const_iterator> std::sregex_token_iterator`
Token iterator for standard strings.

ssub_match

`typedef sub_match<string::const_iterator> std::ssub_match`
Standard regex submatch over a standard string.

wcregex_token_iterator

`typedef regex_token_iterator<const wchar_t*> std::wcregex_token_iterator`
Token iterator for C-style NULL-terminated wide strings.

wcsub_match

`typedef sub_match<const wchar_t*> std::wcsub_match`
Regex submatch over a C-style null-terminated wide string.

wregex

`typedef basic_regex<wchar_t> std::wregex`
Standard wide-character regular expressions.

wsregex_token_iterator

`typedef regex_token_iterator<wstring::const_iterator> std::wsregex_token_iterator`
Token iterator for standard wide-character strings.

wssub_match

```
typedef sub_match<wstring::const_iterator> std::wssub_match
```
Regex submatch over a standard wide string.

3.12.3 Function Documentation

operator<<()

```
template<typename _Ch_type, typename _Ch_traits, typename _Bi_iter>
basic_ostream< _Ch_type, _Ch_traits > & operator<< (
            basic_ostream< _Ch_type, _Ch_traits > & __os,
            const sub_match< _Bi_iter > & __m)  [related]
```
Inserts a matched string into an output stream.

Parameters

__os	The output stream.
__m	A submatch string.

Returns

the output stream with the submatch string inserted.

operator<=>() [1/4]

```
template<typename _Bi_iter, typename _Ch_traits, typename _Alloc>
auto operator<=> (
            const sub_match< _Bi_iter > & __lhs,
            const __sub_match_string< _Bi_iter, _Ch_traits, _Alloc > & __rhs)  [related]
```
Three-way comparison of a regular expression submatch and a string.

Parameters

__lhs	A regular expression submatch.
__rhs	A string.

Returns

A value indicating whether `__lhs` is less than, equal to, greater than, or incomparable with `__rhs`.

operator<=>() [2/4]

```
template<typename _Bi_iter>
auto operator<=> (
            const sub_match< _Bi_iter > & __lhs,
            typename iterator_traits< _Bi_iter >::value_type const & __rhs)  [related]
```
Three-way comparison of a regular expression submatch and a character.

Parameters

__lhs	A regular expression submatch.
__rhs	A character.

Returns

A value indicating whether __lhs is less than, equal to, greater than, or incomparable with __rhs.

operator<=>() [3/4]

```
template<typename _Bi_iter>
auto operator<=> (
            const sub_match< _Bi_iter > & __lhs,
            typename iterator_traits< _Bi_iter >::value_type const * __rhs)  [related]
```

Three-way comparison of a regular expression submatch and a C string.

Parameters

__lhs	A regular expression submatch.
__rhs	A null-terminated string.

Returns

A value indicating whether __lhs is less than, equal to, greater than, or incomparable with __rhs.

operator<=>() [4/4]

```
template<typename _BiIter>
auto operator<=> (
            const sub_match< _BiIter > & __lhs,
            const sub_match< _BiIter > & __rhs)  [related]
```

Three-way comparison of two regular expression submatches.

Parameters

__lhs	First regular expression submatch.
__rhs	Second regular expression submatch.

Returns

A value indicating whether __lhs is less than, equal to, greater than, or incomparable with __rhs.

operator==() [1/5]

```
template<typename _Bi_iter, typename _Alloc>
bool operator== (
            const match_results< _Bi_iter, _Alloc > & __m1,
            const match_results< _Bi_iter, _Alloc > & __m2)  [related]
```

Compares two match_results for equality.

Returns

true if the two objects refer to the same match, false otherwise.

operator==() [2/5]

```
template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc>
bool operator== (
            const sub_match< _Bi_iter > & __lhs,
            const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __rhs)  [related]
```

Tests the equivalence of a regular expression submatch and a string.

Parameters

__lhs	A regular expression submatch.
__rhs	A string.

Returns

true if *__lhs* is equivalent to *__rhs*, false otherwise.

operator==() [3/5]

```
template<typename _Bi_iter>
bool operator== (
            const sub_match< _Bi_iter > & __lhs,
            typename iterator_traits< _Bi_iter >::value_type const & __rhs)  [related]
```

Tests the equivalence of a regular expression submatch and a character.

Parameters

__lhs	A regular expression submatch.
__rhs	A character.

Returns

true if *__lhs* is equivalent to *__rhs*, false otherwise.

operator==() [4/5]

```
template<typename _Bi_iter>
bool operator== (
            const sub_match< _Bi_iter > & __lhs,
            typename iterator_traits< _Bi_iter >::value_type const * __rhs)  [related]
```

Tests the equivalence of a regular expression submatch and a C string.

Parameters

__lhs	A regular expression submatch.
__rhs	A null-terminated string.

Returns

true if *__lhs* is equivalent to *__rhs*, false otherwise.

operator==() [5/5]

```
template<typename _BiIter>
bool operator== (
            const sub_match< _BiIter > & __lhs,
            const sub_match< _BiIter > & __rhs)  [related]
```

Tests the equivalence of two regular expression submatches.

Parameters

__lhs	First regular expression submatch.
__rhs	Second regular expression submatch.

Returns

true if *__lhs* is equivalent to *__rhs*, false otherwise.

regex_match() [1/7]

```
template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>
bool std::regex_match (
            _Bi_iter __first,
            _Bi_iter __last,
            const basic_regex< _Ch_type, _Rx_traits > & __re,
            regex_constants::match_flag_type __flags = regex_constants::match_default)  [inline]
```

Indicates if there is a match between the regular expression `e` and all of the character sequence [first, last).

Parameters

__first	Beginning of the character sequence to match.
__last	One-past-the-end of the character sequence to match.
__re	The regular expression.
__flags	Controls how the regular expression is matched.

Return values

true	A match exists.
false	Otherwise.

Exceptions

an	exception of type regex_error.

References std::regex_constants::match_default, and regex_match().

regex_match() [2/7]

```
template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits>
bool std::regex_match (
            _Bi_iter __s,
```

```
_Bi_iter __e,
match_results< _Bi_iter, _Alloc > & __m,
const basic_regex< _Ch_type, _Rx_traits > & __re,
regex_constants::match_flag_type __flags = regex_constants::match_default)  [inline]
```

Determines if there is a match between the regular expression e and all of the character sequence [first, last).

Parameters

__s	Start of the character sequence to match.
__e	One-past-the-end of the character sequence to match.
__m	The match results.
__re	The regular expression.
__flags	Controls how the regular expression is matched.

Return values

true	A match exists.
false	Otherwise.

Exceptions

an	exception of type regex_error.

References std::regex_constants::match_default.
Referenced by regex_match(), regex_match(), regex_match(), regex_match(), and regex_match().

regex_match() [3/7]

```
template<typename _Ch_type, class _Rx_traits>
bool std::regex_match (
            const _Ch_type * __s,
            const basic_regex< _Ch_type, _Rx_traits > & __re,
            regex_constants::match_flag_type __f = regex_constants::match_default)  [inline]
```

Indicates if there is a match between the regular expression e and a C-style null-terminated string.

Parameters

_↩ _s	The C-style null-terminated string to match.
_↩ _re	The regular expression.
_↩ _f	Controls how the regular expression is matched.

Return values

true	A match exists.
false	Otherwise.

Exceptions

an	exception of type regex_error.

References std::regex_constants::match_default, and regex_match().

regex_match() [4/7]

```
template<typename _Ch_type, typename _Alloc, typename _Rx_traits>
bool std::regex_match (
            const _Ch_type * __s,
            match_results< const _Ch_type *, _Alloc > & __m,
            const basic_regex< _Ch_type, _Rx_traits > & __re,
            regex_constants::match_flag_type __f = regex_constants::match_default)  [inline]
```

Determines if there is a match between the regular expression e and a C-style null-terminated string.

Parameters

_↩ _s	The C-style null-terminated string to match.
_↩ _m	The match results.
_↩ _re	The regular expression.
_↩ _f	Controls how the regular expression is matched.

Return values

true	A match exists.
false	Otherwise.

Exceptions

an	exception of type regex_error.

References std::regex_constants::match_default, and regex_match().

regex_match() [5/7]

```
template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _↩
Rx_traits>
bool std::regex_match (
            const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > && ,
            match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_↩
iterator, _Alloc > & ,
            const basic_regex< _Ch_type, _Rx_traits > & ,
            regex_constants::match_flag_type  = regex_constants::match_default)  [delete]
```

Prevent unsafe attempts to get match_results from a temporary string.
References std::regex_constants::match_default.

regex_match() [6/7]

```
template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _↩
Rx_traits>
bool std::regex_match (
            const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > & __s,
            match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_↩
iterator, _Alloc > & __m,
            const basic_regex< _Ch_type, _Rx_traits > & __re,
            regex_constants::match_flag_type __flags = regex_constants::match_default)  [inline]
```

Determines if there is a match between the regular expression e and a string.

Parameters

__s	The string to match.
__m	The match results.
__re	The regular expression.
__flags	Controls how the regular expression is matched.

Return values

true	A match exists.
false	Otherwise.

Exceptions

an	exception of type regex_error.

References std::basic_string< _CharT, _Traits, _Alloc >::begin(), std::basic_string< _CharT, _Traits, _Alloc >::end(), std::regex_constants::match_default, and regex_match().

regex_match() [7/7]

```
template<typename _Ch_traits, typename _Str_allocator, typename _Ch_type, typename _Rx_traits>
bool std::regex_match (
            const basic_string< _Ch_type, _Ch_traits, _Str_allocator > & __s,
            const basic_regex< _Ch_type, _Rx_traits > & __re,
            regex_constants::match_flag_type __flags = regex_constants::match_default)  [inline]
```

Indicates if there is a match between the regular expression e and a string.

Parameters

__s	[IN] The string to match.
__re	[IN] The regular expression.
__flags	[IN] Controls how the regular expression is matched.

Return values

true	A match exists.
false	Otherwise.

Exceptions

an	exception of type regex_error.

References std::basic_string< _CharT, _Traits, _Alloc >::begin(), std::basic_string< _CharT, _Traits, _Alloc >::end(), std::regex_constants::match_default, and regex_match().

regex_replace() [1/6]

```
template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type>
_Out_iter std::regex_replace (
            _Out_iter __out,
            _Bi_iter __first,
            _Bi_iter __last,
            const basic_regex< _Ch_type, _Rx_traits > & __e,
            const _Ch_type * __fmt,
            regex_constants::match_flag_type __flags = regex_constants::match_default)
```

Search for a regular expression within a range for multiple times, and replace the matched parts through filling a format C-string.

Parameters

__out	[OUT] The output iterator.
__first	[IN] The start of the string to search.
__last	[IN] One-past-the-end of the string to search.
__e	[IN] The regular expression to search for.
__fmt	[IN] The format C-string.
__flags	[IN] Search and replace policy flags.

Returns

__out

Exceptions

an	exception of type regex_error.

References std::regex_constants::match_default.

regex_replace() [2/6]

```
template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type, typename
_St, typename _Sa>
_Out_iter std::regex_replace (
            _Out_iter __out,
            _Bi_iter __first,
            _Bi_iter __last,
            const basic_regex< _Ch_type, _Rx_traits > & __e,
            const basic_string< _Ch_type, _St, _Sa > & __fmt,
            regex_constants::match_flag_type __flags = regex_constants::match_default)  [inline]
```

Search for a regular expression within a range for multiple times, and replace the matched parts through filling a format string.

Parameters

__out	[OUT] The output iterator.
__first	[IN] The start of the string to search.
__last	[IN] One-past-the-end of the string to search.
__e	[IN] The regular expression to search for.
__fmt	[IN] The format string.
__flags	[IN] Search and replace policy flags.

Returns

__out

Exceptions

an	exception of type regex_error.

References std::basic_string< _CharT, _Traits, _Alloc >::c_str(), std::basic_string< _CharT, _Traits, _Alloc >::length(), and std::regex_constants::match_default.
Referenced by regex_replace(), regex_replace(), regex_replace(), and regex_replace().

regex_replace() [3/6]

```
template<typename _Rx_traits, typename _Ch_type>
basic_string< _Ch_type > std::regex_replace (
            const _Ch_type * __s,
            const basic_regex< _Ch_type, _Rx_traits > & __e,
            const _Ch_type * __fmt,
            regex_constants::match_flag_type __flags = regex_constants::match_default)  [inline]
```

Search for a regular expression within a C-string for multiple times, and replace the matched parts through filling a format C-string.

Parameters

__s	[IN] The C-string to search and replace.
__e	[IN] The regular expression to search for.
__fmt	[IN] The format C-string.
__flags	[IN] Search and replace policy flags.

Returns

The string after replacing.

Exceptions

an	exception of type regex_error.

References back_inserter(), std::regex_constants::match_default, and regex_replace().

regex_replace() [4/6]

```
template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa>
basic_string< _Ch_type > std::regex_replace (
            const _Ch_type * __s,
            const basic_regex< _Ch_type, _Rx_traits > & __e,
            const basic_string< _Ch_type, _St, _Sa > & __fmt,
            regex_constants::match_flag_type __flags = regex_constants::match_default)  [inline]
```

Search for a regular expression within a C-string for multiple times, and replace the matched parts through filling a format string.

Parameters

__s	[IN] The C-string to search and replace.
__e	[IN] The regular expression to search for.
__fmt	[IN] The format string.
__flags	[IN] Search and replace policy flags.

Returns

The string after replacing.

Exceptions

an	exception of type regex_error.

References back_inserter(), std::regex_constants::match_default, and regex_replace().

regex_replace() [5/6]

```
template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa>
basic_string< _Ch_type, _St, _Sa > std::regex_replace (
            const basic_string< _Ch_type, _St, _Sa > & __s,
            const basic_regex< _Ch_type, _Rx_traits > & __e,
            const _Ch_type * __fmt,
            regex_constants::match_flag_type __flags = regex_constants::match_default)  [inline]
```

Search for a regular expression within a string for multiple times, and replace the matched parts through filling a format C-string.

Parameters

__s	[IN] The string to search and replace.
__e	[IN] The regular expression to search for.
__fmt	[IN] The format C-string.
__flags	[IN] Search and replace policy flags.

Returns

The string after replacing.

Exceptions

an	exception of type regex_error.

References back_inserter(), std::basic_string< _CharT, _Traits, _Alloc >::begin(), std::basic_string< _CharT, _Traits, _Alloc >::end(), std::regex_constants::match_default, and regex_replace().

regex_replace() [6/6]

```
template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa, typename _Fst, typename
_Fsa>
basic_string< _Ch_type, _St, _Sa > std::regex_replace (
            const basic_string< _Ch_type, _St, _Sa > & __s,
            const basic_regex< _Ch_type, _Rx_traits > & __e,
            const basic_string< _Ch_type, _Fst, _Fsa > & __fmt,
            regex_constants::match_flag_type __flags = regex_constants::match_default)  [inline]
```

Search for a regular expression within a string for multiple times, and replace the matched parts through filling a format string.

Parameters

__s	[IN] The string to search and replace.
__e	[IN] The regular expression to search for.
__fmt	[IN] The format string.
__flags	[IN] Search and replace policy flags.

Returns

The string after replacing.

Exceptions

an	exception of type regex_error.

References back_inserter(), std::basic_string< _CharT, _Traits, _Alloc >::begin(), std::basic_string< _CharT, _Traits, _Alloc >::end(), std::regex_constants::match_default, and regex_replace().

regex_search() [1/7]

```
template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>
bool std::regex_search (
            _Bi_iter __first,
            _Bi_iter __last,
            const basic_regex< _Ch_type, _Rx_traits > & __re,
            regex_constants::match_flag_type __flags = regex_constants::match_default)  [inline]
```

Searches for a regular expression within a range.

Parameters

__first	[IN] The start of the string to search.
__last	[IN] One-past-the-end of the string to search.
__re	[IN] The regular expression to search for.

__flags	[IN] Search policy flags.

Return values

true	A match was found within the string.
false	No match was found within the string.

Exceptions

an	exception of type regex_error.

References std::regex_constants::match_default, and regex_search().

regex_search() [2/7]

```
template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits>
bool std::regex_search (
             _Bi_iter __s,
             _Bi_iter __e,
             match_results< _Bi_iter, _Alloc > & __m,
             const basic_regex< _Ch_type, _Rx_traits > & __re,
             regex_constants::match_flag_type __flags = regex_constants::match_default)  [inline]
```

Searches for a regular expression within a range.

Parameters

__s	[IN] The start of the string to search.
__e	[IN] One-past-the-end of the string to search.
__m	[OUT] The match results.
__re	[IN] The regular expression to search for.
__flags	[IN] Search policy flags.

Return values

true	A match was found within the string.
false	No match was found within the string, the content of m is undefined.

Exceptions

an	exception of type regex_error.

References std::regex_constants::match_default.
Referenced by std::regex_iterator< const char * >::regex_iterator(), std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++(), regex_search(), regex_search(), regex_search(), regex_search(), and regex_search().

regex_search() [3/7]

```
template<typename _Ch_type, typename _Rx_traits>
bool std::regex_search (
             const _Ch_type * __s,
             const basic_regex< _Ch_type, _Rx_traits > & __e,
             regex_constants::match_flag_type __f = regex_constants::match_default)  [inline]
```

Searches for a regular expression within a C-string.

Parameters

_↩ _s	[IN] The C-string to search.
_↩ _e	[IN] The regular expression to search for.
_↩ _f	[IN] Search policy flags.

Return values

true	A match was found within the string.
false	No match was found within the string.

Exceptions

an	exception of type regex_error.

References std::regex_constants::match_default, and regex_search().

regex_search() [4/7]

```
template<typename _Ch_type, class _Alloc, class _Rx_traits>
bool std::regex_search (
             const _Ch_type * __s,
             match_results< const _Ch_type *, _Alloc > & __m,
             const basic_regex< _Ch_type, _Rx_traits > & __e,
             regex_constants::match_flag_type __f = regex_constants::match_default)  [inline]
```

Searches for a regular expression within a C-string.

Parameters

_↩ _s	[IN] A C-string to search for the regex.
_↩ _m	[OUT] The set of regex matches.
_↩ _e	[IN] The regex to search for in s.
_↩ _f	[IN] The search flags.

Return values

true	A match was found within the string.
false	No match was found within the string, the content of m is undefined.

Exceptions

an	exception of type regex_error.

References std::regex_constants::match_default, and regex_search().

regex_search() [5/7]

```
template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _↩
Rx_traits>
bool std::regex_search (
            const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > && ,
            match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_↩
iterator, _Alloc > & ,
            const basic_regex< _Ch_type, _Rx_traits > & ,
            regex_constants::match_flag_type  = regex_constants::match_default)  [delete]
```

Prevent unsafe attempts to get match_results from a temporary string.
References std::regex_constants::match_default.

regex_search() [6/7]

```
template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _↩
Rx_traits>
bool std::regex_search (
            const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > & __s,
            match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_↩
iterator, _Alloc > & __m,
            const basic_regex< _Ch_type, _Rx_traits > & __e,
            regex_constants::match_flag_type __f = regex_constants::match_default)  [inline]
```

Searches for a regular expression within a string.

Parameters

_↩ _*s*	[IN] A C++ string to search for the regex.
_↩ _*m*	[OUT] The set of regex matches.
_↩ _*e*	[IN] The regex to search for in s.
_↩ _*f*	[IN] The search flags.

Return values

true	A match was found within the string.
false	No match was found within the string, the content of m is undefined.

Exceptions

an	exception of type regex_error.

References std::basic_string< _CharT, _Traits, _Alloc >::begin(), std::basic_string< _CharT, _Traits, _Alloc >::end(), std::regex_constants::match_default, and regex_search().

regex_search() [7/7]

```
template<typename _Ch_traits, typename _String_allocator, typename _Ch_type, typename _Rx_traits>
bool std::regex_search (
            const basic_string< _Ch_type, _Ch_traits, _String_allocator > & __s,
            const basic_regex< _Ch_type, _Rx_traits > & __e,
            regex_constants::match_flag_type __flags = regex_constants::match_default)  [inline]
```

Searches for a regular expression within a string.

Parameters

__s	[IN] The string to search.
__e	[IN] The regular expression to search for.
__flags	[IN] Search policy flags.

Return values

true	A match was found within the string.
false	No match was found within the string.

Exceptions

an	exception of type regex_error.

References std::regex_constants::match_default, and regex_search().

swap() [1/2]

```
template<typename _Ch_type, typename _Rx_traits>
void swap (
            basic_regex< _Ch_type, _Rx_traits > & __lhs,
            basic_regex< _Ch_type, _Rx_traits > & __rhs)  [related]
```

Swaps the contents of two regular expression objects.

Parameters

__lhs	First regular expression.
__rhs	Second regular expression.

swap() [2/2]

```
template<typename _Bi_iter, typename _Alloc>
void swap (
            match_results< _Bi_iter, _Alloc > & __lhs,
            match_results< _Bi_iter, _Alloc > & __rhs)  [related]
```

Swaps two match results.

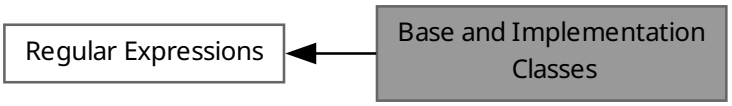
Parameters

__lhs	A match result.
__rhs	A match result.

The contents of the two match_results objects are swapped.

3.12.4 Base and Implementation Classes

Collaboration diagram for Base and Implementation Classes:



Classes

- struct std::__detail::_BracketMatcher< _TraitsT, __icase, __collate >
- class std::__detail::_Compiler< _TraitsT >
- class std::__detail::_Executor< _BiIter, _Alloc, _TraitsT, __dfs_mode >
- class std::__detail::_Scanner< _CharT >
- class std::__detail::_StateSeq< _TraitsT >

Typedefs

- template<typename _CharT>
  using **std::__detail::_Matcher**
- typedef long **std::__detail::_StateIdT**

Enumerations

- enum std::__detail::_Opcode : int {
  _S_opcode_unknown , **_S_opcode_alternative** , **_S_opcode_repeat** , **_S_opcode_backref** ,
  _S_opcode_line_begin_assertion , **_S_opcode_line_end_assertion** , **_S_opcode_word_boundary** , **_S_↩**
  opcode_subexpr_lookahead ,
  _S_opcode_subexpr_begin , **_S_opcode_subexpr_end** , **_S_opcode_dummy** , **_S_opcode_match** ,
  _S_opcode_accept }

Variables

- constexpr _StateIdT **std::__detail::_S_invalid_state_id**

3.12.4.1 Detailed Description

3.12.4.2 Enumeration Type Documentation

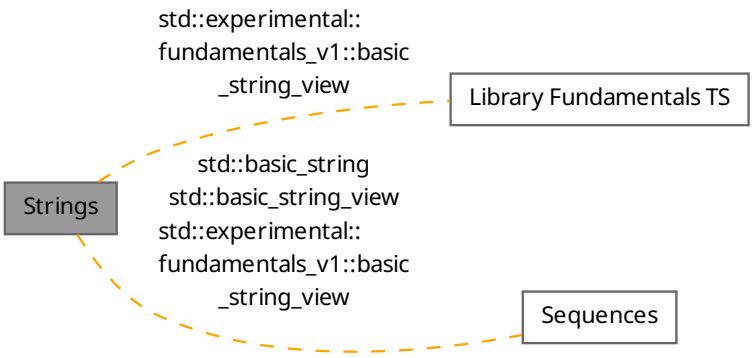
_Opcode

```
enum std::__detail::_Opcode :  int
```

Operation codes that define the type of transitions within the base NFA that represents the regular expression.

3.13 Strings

Collaboration diagram for Strings:



Classes

- class std::basic_string< _CharT, _Traits, _Alloc >
- class std::basic_string_view< _CharT, _Traits >
- class std::experimental::fundamentals_v1::basic_string_view< _CharT, _Traits >
- struct std::char_traits< _CharT >

Typedefs

- typedef basic_string< char > std::string
- typedef basic_string< char16_t > std::u16string
- typedef basic_string< char32_t > std::u32string
- typedef basic_string< wchar_t > std::wstring

3.13.1 Detailed Description

3.13.2 Typedef Documentation

string

```
typedef basic_string<char> std::string
```

A string of `char`.

u16string

```
typedef basic_string<char16_t> std::u16string
```

A string of `char16_t`.

u32string

```
typedef basic_string<char32_t> std::u32string
```

A string of `char32_t`.

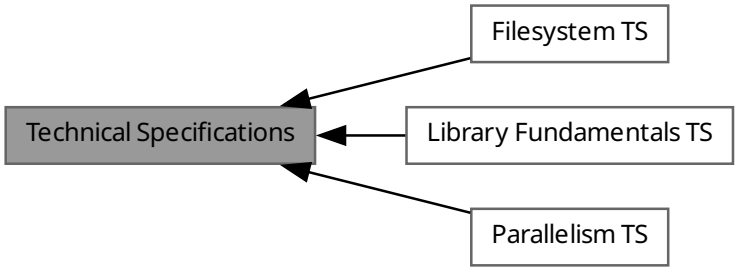
wstring

```
typedef basic_string<wchar_t> std::wstring
```

A string of `wchar_t`.

3.14 Technical Specifications

Collaboration diagram for Technical Specifications:



Topics

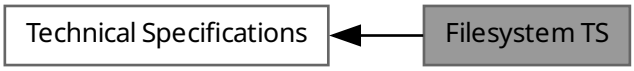
- Filesystem TS
- Library Fundamentals TS
- Parallelism TS

3.14.1 Detailed Description

Components specified by various Technical Specifications.
As indicated by the std::experimental namespace and the header paths, the contents of these Technical Specifications are experimental and not part of the C++ standard. As such the interfaces and implementations may change in the future, and there is **no guarantee of compatibility between different GCC releases** for these features.

3.14.2 Filesystem TS

Collaboration diagram for Filesystem TS:



Files

- file filesystem

Classes

- class std::experimental::filesystem::v1::filesystem_error
- class std::experimental::filesystem::v1::path::iterator
- class std::experimental::filesystem::v1::path
- struct std::experimental::filesystem::v1::space_info

Typedefs

- using std::experimental::filesystem::file_time_type

Enumerations

- enum class std::experimental::filesystem::copy_options : unsigned short {
  none , **skip_existing** , **overwrite_existing** , **update_existing** ,
  recursive , **copy_symlinks** , **skip_symlinks** , **directories_only** ,
  create_symlinks , **create_hard_links** }
- enum class std::experimental::filesystem::directory_options : unsigned char { **none** , **follow_directory_symlink** , **skip_permission_denied** }
- enum class std::experimental::filesystem::file_type : signed char {
  none , **not_found** , **regular** , **directory** ,
  symlink , **block** , **character** , **fifo** ,
  socket , **unknown** }
- enum class std::experimental::filesystem::perms : unsigned {
  none , **owner_read** , **owner_write** , **owner_exec** ,
  owner_all , **group_read** , **group_write** , **group_exec** ,
  group_all , **others_read** , **others_write** , **others_exec** ,
  others_all , **all** , **set_uid** , **set_gid** ,
  sticky_bit , **mask** , **unknown** , **add_perms** ,
  remove_perms , **symlink_nofollow** }

Functions

- **std::experimental::filesystem::v1::path::path** (const path &__p)
- **std::experimental::filesystem::v1::path::path** (path &&__p) noexcept
- **std::experimental::filesystem::v1::path::path** (string_type &&__source)
- path **std::experimental::filesystem::absolute** (const path &__p, const path &__base=current_path())
- path & **std::experimental::filesystem::v1::path::assign** (string_type &&__source)
- directory_iterator **std::experimental::filesystem::begin** (directory_iterator __iter) noexcept
- recursive_directory_iterator **std::experimental::filesystem::begin** (recursive_directory_iterator __iter) noexcept
- iterator **std::experimental::filesystem::v1::path::begin** () const noexcept
- path **std::experimental::filesystem::canonical** (const path &__p, const path &__base, error_code &__ec)
- path **std::experimental::filesystem::canonical** (const path &__p, const path &__base=current_path())
- path **std::experimental::filesystem::canonical** (const path &__p, error_code &__ec)
- int **std::experimental::filesystem::v1::path::compare** (const basic_string_view< value_type > __s) const
- int **std::experimental::filesystem::v1::path::compare** (const string_type &__s) const
- int **std::experimental::filesystem::v1::path::compare** (const value_type *__s) const
- void **std::experimental::filesystem::copy** (const path &__from, const path &__to)
- void **std::experimental::filesystem::copy** (const path &__from, const path &__to, copy_options __options)
- void **std::experimental::filesystem::copy** (const path &__from, const path &__to, copy_options __options, error_code &) noexcept
- void **std::experimental::filesystem::copy** (const path &__from, const path &__to, error_code &__ec) noexcept
- bool **std::experimental::filesystem::copy_file** (const path &__from, const path &__to)

- bool **std::experimental::filesystem::copy_file** (const path &__from, const path &__to, copy_options __option)
- bool **std::experimental::filesystem::copy_file** (const path &__from, const path &__to, copy_options __option, error_code &)
- bool **std::experimental::filesystem::copy_file** (const path &__from, const path &__to, error_code &__ec)
- void **std::experimental::filesystem::copy_symlink** (const path &__existing_symlink, const path &__new_↩ symlink)
- void **std::experimental::filesystem::copy_symlink** (const path &__existing_symlink, const path &__new_↩ symlink, error_code &__ec) noexcept
- bool **std::experimental::filesystem::create_directories** (const path &__p)
- bool **std::experimental::filesystem::create_directories** (const path &__p, error_code &__ec)
- bool **std::experimental::filesystem::create_directory** (const path &__p)
- bool **std::experimental::filesystem::create_directory** (const path &__p, const path &__attributes)
- bool **std::experimental::filesystem::create_directory** (const path &__p, const path &__attributes, error_code &__ec) noexcept
- bool **std::experimental::filesystem::create_directory** (const path &__p, error_code &__ec) noexcept
- void **std::experimental::filesystem::create_directory_symlink** (const path &__to, const path &__new_symlink)
- void **std::experimental::filesystem::create_directory_symlink** (const path &__to, const path &__new_symlink, error_code &__ec) noexcept
- void **std::experimental::filesystem::create_hard_link** (const path &__to, const path &__new_hard_link)
- void **std::experimental::filesystem::create_hard_link** (const path &__to, const path &__new_hard_link, error_code &__ec) noexcept
- void **std::experimental::filesystem::create_symlink** (const path &__to, const path &__new_symlink)
- void **std::experimental::filesystem::create_symlink** (const path &__to, const path &__new_symlink, error_code &__ec) noexcept
- path **std::experimental::filesystem::current_path** ()
- void **std::experimental::filesystem::current_path** (const path &__p)
- void **std::experimental::filesystem::current_path** (const path &__p, error_code &__ec) noexcept
- path **std::experimental::filesystem::current_path** (error_code &__ec)
- directory_iterator **std::experimental::filesystem::end** (directory_iterator) noexcept
- recursive_directory_iterator **std::experimental::filesystem::end** (recursive_directory_iterator) noexcept
- iterator **std::experimental::filesystem::v1::path::end** () const noexcept
- bool **std::experimental::filesystem::equivalent** (const path &__p1, const path &__p2)
- bool **std::experimental::filesystem::equivalent** (const path &__p1, const path &__p2, error_code &__ec) noexcept
- bool **std::experimental::filesystem::exists** (const path &__p)
- bool **std::experimental::filesystem::exists** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::exists** (file_status __s) noexcept
- path **std::experimental::filesystem::v1::path::extension** () const
- uintmax_t **std::experimental::filesystem::file_size** (const path &__p)
- uintmax_t **std::experimental::filesystem::file_size** (const path &__p, error_code &__ec) noexcept
- path **std::experimental::filesystem::v1::path::filename** () const
- std::string **std::experimental::filesystem::v1::path::generic_string** () const
- template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>> std::basic_string< _CharT, _Traits, _Allocator > **std::experimental::filesystem::v1::path::generic_string** (const _Allocator &__a=_Allocator()) const
- std::u16string **std::experimental::filesystem::v1::path::generic_u16string** () const
- std::u32string **std::experimental::filesystem::v1::path::generic_u32string** () const
- std::string **std::experimental::filesystem::v1::path::generic_u8string** () const
- std::wstring **std::experimental::filesystem::v1::path::generic_wstring** () const
- uintmax_t **std::experimental::filesystem::hard_link_count** (const path &__p)
- uintmax_t **std::experimental::filesystem::hard_link_count** (const path &__p, error_code &__ec) noexcept

- bool **std::experimental::filesystem::v1::path::has_extension** () const
- bool **std::experimental::filesystem::v1::path::has_stem** () const
- bool **std::experimental::filesystem::v1::path::is_absolute** () const
- bool **std::experimental::filesystem::is_block_file** (const path &__p)
- bool **std::experimental::filesystem::is_block_file** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::is_block_file** (file_status __s) noexcept
- bool **std::experimental::filesystem::is_character_file** (const path &__p)
- bool **std::experimental::filesystem::is_character_file** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::is_character_file** (file_status __s) noexcept
- bool **std::experimental::filesystem::is_directory** (const path &__p)
- bool **std::experimental::filesystem::is_directory** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::is_directory** (file_status __s) noexcept
- bool **std::experimental::filesystem::is_empty** (const path &__p)
- bool **std::experimental::filesystem::is_empty** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::is_fifo** (const path &__p)
- bool **std::experimental::filesystem::is_fifo** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::is_fifo** (file_status __s) noexcept
- bool **std::experimental::filesystem::is_other** (const path &__p)
- bool **std::experimental::filesystem::is_other** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::is_other** (file_status __s) noexcept
- bool **std::experimental::filesystem::is_regular_file** (const path &__p)
- bool **std::experimental::filesystem::is_regular_file** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::is_regular_file** (file_status) noexcept
- bool **std::experimental::filesystem::is_socket** (const path &__p)
- bool **std::experimental::filesystem::is_socket** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::is_socket** (file_status __s) noexcept
- bool **std::experimental::filesystem::is_symlink** (const path &__p)
- bool **std::experimental::filesystem::is_symlink** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::is_symlink** (file_status) noexcept
- file_time_type **std::experimental::filesystem::last_write_time** (const path &__p)
- file_time_type **std::experimental::filesystem::last_write_time** (const path &__p, error_code &__ec) noexcept
- void **std::experimental::filesystem::last_write_time** (const path &__p, file_time_type __new_time)
- void **std::experimental::filesystem::last_write_time** (const path &__p, file_time_type __new_time, error_code &__ec) noexcept
- path & **std::experimental::filesystem::v1::path::make_preferred** ()
- bool **std::experimental::filesystem::operator!=** (const directory_iterator &__lhs, const directory_iterator &__↩ rhs)
- bool **std::experimental::filesystem::operator!=** (const recursive_directory_iterator &__lhs, const recursive_↩ directory_iterator &__rhs)
- copy_options & **std::experimental::filesystem::operator&=** (copy_options &__x, copy_options __y) noexcept
- reference **std::experimental::filesystem::v1::path::iterator::operator*** () const noexcept
- iterator & **std::experimental::filesystem::v1::path::iterator::operator++** () noexcept
- template<typename _CharT>
  __detail::_Path< _CharT *, _CharT * > & **std::experimental::filesystem::v1::path::operator+=** (_CharT __x)
- path & **std::experimental::filesystem::v1::path::operator+=** (basic_string_view< value_type > __x)
- path & **std::experimental::filesystem::v1::path::operator+=** (const path &__x)
- path & **std::experimental::filesystem::v1::path::operator+=** (const string_type &__x)
- path & **std::experimental::filesystem::v1::path::operator+=** (const value_type *__x)
- path & **std::experimental::filesystem::v1::path::operator+=** (value_type __x)
- iterator & **std::experimental::filesystem::v1::path::iterator::operator--** () noexcept
- bool **std::experimental::filesystem::operator<** (const path &__lhs, const path &__rhs) noexcept

- path & **std::experimental::filesystem::v1::path::operator=** (const path &__p)
- path & **std::experimental::filesystem::v1::path::operator=** (path &&__p) noexcept
- path & **std::experimental::filesystem::v1::path::operator=** (string_type &&__source)
- bool **std::experimental::filesystem::operator==** (const directory_iterator &__lhs, const directory_iterator &_↩ _rhs)
- bool **std::experimental::filesystem::operator==** (const path &__lhs, const path &__rhs) noexcept
- bool **std::experimental::filesystem::operator==** (const recursive_directory_iterator &__lhs, const recursive_↩ directory_iterator &__rhs)
- constexpr copy_options **std::experimental::filesystem::operator^** (copy_options __x, copy_options __y) noexcept
- copy_options & **std::experimental::filesystem::operator^=** (copy_options &__x, copy_options __y) noexcept
- constexpr copy_options **std::experimental::filesystem::operator|** (copy_options __x, copy_options __y) noexcept
- copy_options & **std::experimental::filesystem::operator|=** (copy_options &__x, copy_options __y) noexcept
- constexpr copy_options **std::experimental::filesystem::operator~** (copy_options __x) noexcept
- void **std::experimental::filesystem::permissions** (const path &__p, perms __prms)
- void **std::experimental::filesystem::permissions** (const path &__p, perms __prms, error_code &__ec) noexcept
- path **std::experimental::filesystem::read_symlink** (const path &__p)
- path **std::experimental::filesystem::read_symlink** (const path &__p, error_code &__ec)
- bool **std::experimental::filesystem::remove** (const path &__p)
- bool **std::experimental::filesystem::remove** (const path &__p, error_code &__ec) noexcept
- uintmax_t **std::experimental::filesystem::remove_all** (const path &__p)
- uintmax_t **std::experimental::filesystem::remove_all** (const path &__p, error_code &__ec)
- void **std::experimental::filesystem::rename** (const path &__from, const path &__to)
- void **std::experimental::filesystem::rename** (const path &__from, const path &__to, error_code &__ec) noexcept
- void **std::experimental::filesystem::resize_file** (const path &__p, uintmax_t __size)
- void **std::experimental::filesystem::resize_file** (const path &__p, uintmax_t __size, error_code &__ec) noexcept
- space_info **std::experimental::filesystem::space** (const path &__p)
- space_info **std::experimental::filesystem::space** (const path &__p, error_code &__ec) noexcept
- file_status **std::experimental::filesystem::status** (const path &)
- file_status **std::experimental::filesystem::status** (const path &, error_code &) noexcept
- bool **std::experimental::filesystem::status_known** (file_status) noexcept
- path **std::experimental::filesystem::v1::path::stem** () const
- std::string **std::experimental::filesystem::v1::path::string** () const
- template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>> std::basic_string< _CharT, _Traits, _Allocator > **std::experimental::filesystem::v1::path::string** (const _↩ Allocator &__a=_Allocator()) const
- void **std::experimental::filesystem::v1::path::swap** (path &__rhs) noexcept
- file_status **std::experimental::filesystem::symlink_status** (const path &)
- file_status **std::experimental::filesystem::symlink_status** (const path &, error_code &) noexcept
- path **std::experimental::filesystem::system_complete** (const path &__p)
- path **std::experimental::filesystem::system_complete** (const path &__p, error_code &__ec)
- path **std::experimental::filesystem::temp_directory_path** ()
- path **std::experimental::filesystem::temp_directory_path** (error_code &__ec)
- std::u16string **std::experimental::filesystem::v1::path::u16string** () const
- std::u32string **std::experimental::filesystem::v1::path::u32string** () const
- std::string **std::experimental::filesystem::v1::path::u8string** () const
- std::wstring **std::experimental::filesystem::v1::path::wstring** () const

- constexpr perms **std::experimental::filesystem::operator|** (perms __x, perms __y) noexcept
- constexpr perms **std::experimental::filesystem::operator^** (perms __x, perms __y) noexcept
- constexpr perms **std::experimental::filesystem::operator~** (perms __x) noexcept
- perms & **std::experimental::filesystem::operator&=** (perms &__x, perms __y) noexcept
- perms & **std::experimental::filesystem::operator|=** (perms &__x, perms __y) noexcept
- perms & **std::experimental::filesystem::operator^=** (perms &__x, perms __y) noexcept

- constexpr directory_options **std::experimental::filesystem::operator|** (directory_options __x, directory_options __y) noexcept
- constexpr directory_options **std::experimental::filesystem::operator^** (directory_options __x, directory_options __y) noexcept
- constexpr directory_options **std::experimental::filesystem::operator~** (directory_options __x) noexcept
- directory_options & **std::experimental::filesystem::operator&=** (directory_options &__x, directory_options __y) noexcept
- directory_options & **std::experimental::filesystem::operator|=** (directory_options &__x, directory_options __y) noexcept
- directory_options & **std::experimental::filesystem::operator^=** (directory_options &__x, directory_options __y) noexcept

3.14.2.1 Detailed Description

Utilities for performing operations on file systems and their components, such as paths, regular files, and directories.
ISO/IEC TS 18822:2015 C++ File System Technical Specification

Since

C++11

Remarks

Link using `-lstdc++fs` to use these types and functions.

3.14.2.2 Typedef Documentation

file_time_type

```
using std::experimental::filesystem::v1::file_time_type
```
The type used for file timestamps.

3.14.2.3 Enumeration Type Documentation

copy_options

```
enum class std::experimental::filesystem::v1::copy_options : unsigned short [strong]
```
Bitmask type controlling effects of `filesystem::copy`

directory_options

```
enum class std::experimental::filesystem::v1::directory_options : unsigned char [strong]
```
Bitmask type controlling directory iteration.

file_type

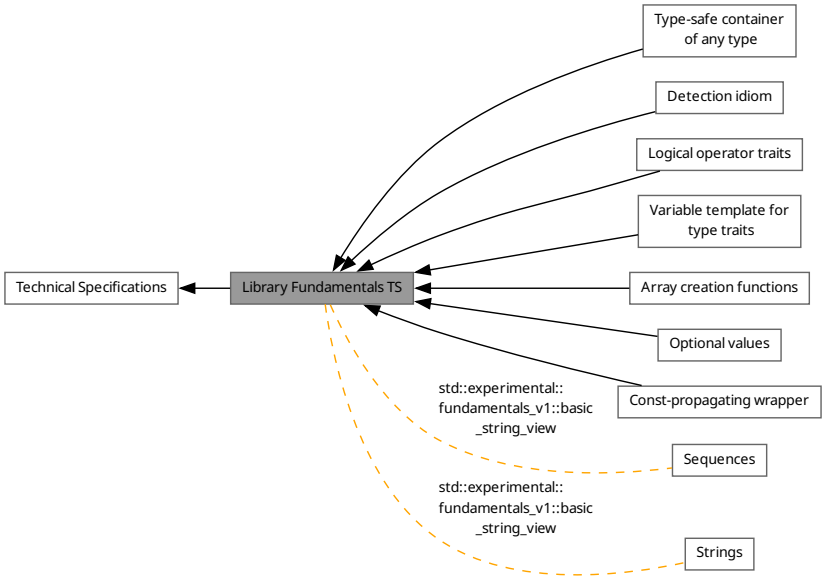
```
enum class std::experimental::filesystem::v1::file_type : signed char [strong]
```
Enumerated type representing the type of a file.

perms

`enum class std::experimental::filesystem::v1::perms : unsigned [strong]`
Bitmask type representing file access permissions.

3.14.3 Library Fundamentals TS

Collaboration diagram for Library Fundamentals TS:



Topics

- Array creation functions
- Const-propagating wrapper
- Detection idiom
- Logical operator traits
- Optional values
- Type-safe container of any type
- Variable template for type traits

Files

- file algorithm
- file any
- file array
- file chrono
- file deque
- file forward_list
- file functional
- file iterator
- file list

- file map
- file memory
- file memory_resource
- file numeric
- file optional
- file propagate_const
- file random
- file ratio
- file regex
- file set
- file string
- file string_view
- file system_error
- file tuple
- file type_traits
- file unordered_map
- file unordered_set
- file utility
- file vector

Classes

- class std::experimental::fundamentals_v1::basic_string_view< _CharT, _Traits >

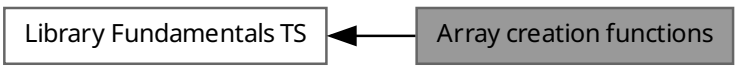
3.14.3.1 Detailed Description

Components defined by the *C++ Extensions for Library Fundamentals* Technical Specification, versions 1 and 2.

- ISO/IEC TS 19568:2015 C++ Extensions for Library Fundamentals
- ISO/IEC TS 19568:2017 C++ Extensions for Library Fundamentals, Version 2

3.14.3.2 Array creation functions

Collaboration diagram for Array creation functions:



Functions

- template<typename _Tp, size_t _Nm, size_t... _Idx>
  constexpr array< remove_cv_t< _Tp >, _Nm > **std::experimental::__to_array** (_Tp(&__a)[_Nm], index_sequence< _Idx... >)
- template<typename _Dest = void, typename... _Types>
  constexpr array< typename __make_array_elem< _Dest, _Types... >::type, sizeof...(_Types)> std::experimental::make_array (_Types &&... __t)

- template<typename _Tp, size_t _Nm>
  constexpr array< remove_cv_t< _Tp >, _Nm > std::experimental::to_array (_Tp(&__a)[_Nm]) noexcept(is_nothrow_constructible< remove_cv_t< _Tp >, _Tp & >::value)

3.14.3.2.1 Detailed Description

Array creation functions as described in N4529, Working Draft, C++ Extensions for Library Fundamentals, Version 2

3.14.3.2.2 Function Documentation

make_array()

```
template<typename _Dest = void, typename...  _Types>
array< typename __make_array_elem< _Dest, _Types...  >::type, sizeof...(_Types)> std::experimental↩
::fundamentals_v2::make_array (
            _Types &&...  __t)  [constexpr]
```

Create a std::array from a variable-length list of arguments.

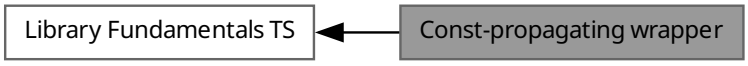
to_array()

```
template<typename _Tp, size_t _Nm>
array< remove_cv_t< _Tp >, _Nm > std::experimental::fundamentals_v2::to_array (
            _Tp(&) __a[_Nm])  [constexpr], [noexcept]
```

Create a std::array from an array.

3.14.3.3 Const-propagating wrapper

Collaboration diagram for Const-propagating wrapper:



Classes

- class std::experimental::fundamentals_v2::propagate_const< _Tp >

Functions

- template<typename _Tp>
  constexpr const _Tp & **std::experimental::get_underlying** (const propagate_const< _Tp > &__pt) noexcept
- template<typename _Tp>
  constexpr _Tp & **std::experimental::get_underlying** (propagate_const< _Tp > &__pt) noexcept
- constexpr **std::experimental::fundamentals_v2::__propagate_const_conversion_nc< _Tp, _Elem, true >::operator _Elem** * ()
- constexpr **std::experimental::fundamentals_v2::__propagate_const_conversions< _Tp * >::operator _Tp** * () noexcept
- constexpr **std::experimental::fundamentals_v2::__propagate_const_conversion_c< _Tp, _Elem, true >↩ ::operator const _Elem** * () const

- constexpr **std::experimental::fundamentals_v2::__propagate_const_conversions**< **_Tp** * >**::operator const _Tp** * () const noexcept
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator!=** (const _Tp &__t, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator!=** (const propagate_const< _Tp > &__pt, const _Up &__u)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator!=** (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)
- template<typename _Tp>
  constexpr bool **std::experimental::operator!=** (const propagate_const< _Tp > &__pt, nullptr_t)
- template<typename _Tp>
  constexpr bool **std::experimental::operator!=** (nullptr_t, const propagate_const< _Tp > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator**< (const _Tp &__t, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator**< (const propagate_const< _Tp > &__pt, const _Up &__u)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator**< (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator**<= (const _Tp &__t, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator**<= (const propagate_const< _Tp > &__pt, const _Up &__u)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator**<= (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator==** (const _Tp &__t, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator==** (const propagate_const< _Tp > &__pt, const _Up &__u)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator==** (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)
- template<typename _Tp>
  constexpr bool **std::experimental::operator==** (const propagate_const< _Tp > &__pt, nullptr_t)
- template<typename _Tp>
  constexpr bool **std::experimental::operator==** (nullptr_t, const propagate_const< _Tp > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator**> (const _Tp &__t, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator**> (const propagate_const< _Tp > &__pt, const _Up &__u)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator**> (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator**>= (const _Tp &__t, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator**>= (const propagate_const< _Tp > &__pt, const _Up &__u)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator**>= (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)

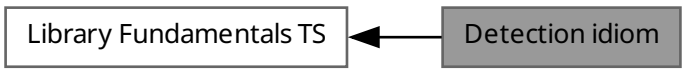
- template<typename _Tp>
  constexpr enable_if_t< __is_swappable< _Tp >::value, void > **std::experimental::swap** (propagate_const< _Tp > &__pt, propagate_const< _Tp > &__pt2) noexcept(__is_nothrow_swappable< _Tp >::value)

3.14.3.3.1 Detailed Description

A const-propagating wrapper that propagates const to pointer-like members, as described in n4388 "A Proposal to Add a Const-Propagating Wrapper to the Standard Library".

3.14.3.4 Detection idiom

Collaboration diagram for Detection idiom:



- #define __cpp_lib_experimental_detect
- template<typename...>
  using std::experimental::void_t
- template<typename _Default, template< typename... > class _Op, typename... _Args>
  using std::experimental::detected_or
- template<typename _Default, template< typename... > class _Op, typename... _Args>
  using std::experimental::detected_or_t
- template<template< typename... > class _Op, typename... _Args>
  using std::experimental::detected_t
- template<template< typename... > class _Op, typename... _Args>
  using std::experimental::is_detected
- template<typename _Expected, template< typename... > class _Op, typename... _Args>
  using std::experimental::is_detected_exact
- template<typename _To, template< typename... > class _Op, typename... _Args>
  using std::experimental::is_detected_convertible
- template<template< typename... > class _Op, typename... _Args>
  constexpr bool std::experimental::is_detected_v
- template<typename _Expected, template< typename... > class _Op, typename... _Args>
  constexpr bool std::experimental::is_detected_exact_v
- template<typename _To, template< typename... > class _Op, typename... _Args>
  constexpr bool std::experimental::is_detected_convertible_v

3.14.3.4.1 Detailed Description

Since

Library Fundamentals TS v2. C++14.

3.14.3.4.2 Macro Definition Documentation

__cpp_lib_experimental_detect

```
#define __cpp_lib_experimental_detect
```

A metafunction that always yields void, used for detecting valid types.

3.14.3.4.3 Typedef Documentation

detected_or

```
template<typename _Default, template< typename... > class _Op, typename... _Args>
using std::experimental::fundamentals_v2::detected_or
```

A metafunction that always yields void, used for detecting valid types.

detected_or_t

```
template<typename _Default, template< typename... > class _Op, typename... _Args>
using std::experimental::fundamentals_v2::detected_or_t
```

A metafunction that always yields void, used for detecting valid types.

detected_t

```
template<template< typename... > class _Op, typename... _Args>
using std::experimental::fundamentals_v2::detected_t
```

A metafunction that always yields void, used for detecting valid types.

is_detected

```
template<template< typename... > class _Op, typename... _Args>
using std::experimental::fundamentals_v2::is_detected
```

A metafunction that always yields void, used for detecting valid types.

is_detected_convertible

```
template<typename _To, template< typename... > class _Op, typename... _Args>
using std::experimental::fundamentals_v2::is_detected_convertible
```

A metafunction that always yields void, used for detecting valid types.

is_detected_exact

```
template<typename _Expected, template< typename... > class _Op, typename... _Args>
using std::experimental::fundamentals_v2::is_detected_exact
```

A metafunction that always yields void, used for detecting valid types.

void_t

```
template<typename...>
using std::experimental::fundamentals_v2::void_t
```

A metafunction that always yields void, used for detecting valid types.

3.14.3.4.4 Variable Documentation

is_detected_convertible_v

```
template<typename _To, template< typename... > class _Op, typename... _Args>
bool std::experimental::fundamentals_v2::is_detected_convertible_v [constexpr]
```

A metafunction that always yields void, used for detecting valid types.

is_detected_exact_v

```
template<typename _Expected, template< typename... > class _Op, typename... _Args>
bool std::experimental::fundamentals_v2::is_detected_exact_v [constexpr]
```

A metafunction that always yields void, used for detecting valid types.

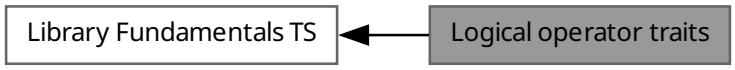
is_detected_v

```
template<template< typename... > class _Op, typename... _Args>
bool std::experimental::fundamentals_v2::is_detected_v [constexpr]
```

A metafunction that always yields void, used for detecting valid types.

3.14.3.5 Logical operator traits

Collaboration diagram for Logical operator traits:



- #define **__cpp_lib_experimental_logical_traits**
- template<typename... _Bn>
  constexpr bool **std::experimental::conjunction_v**
- template<typename... _Bn>
  constexpr bool **std::experimental::disjunction_v**
- template<typename _Pp>
  constexpr bool **std::experimental::negation_v**

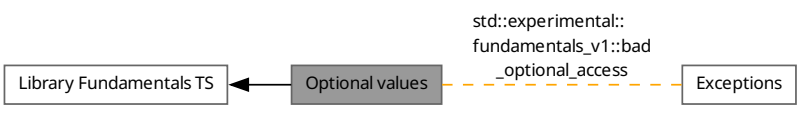
3.14.3.5.1 Detailed Description

Since

Library Fundamentals TS v2. C++14.

3.14.3.6 Optional values

Collaboration diagram for Optional values:



Classes

- class std::experimental::fundamentals_v1::bad_optional_access
- struct std::experimental::fundamentals_v1::in_place_t
- struct std::experimental::fundamentals_v1::nullopt_t
- class std::experimental::fundamentals_v1::optional< _Tp >

Macros

- #define **__cpp_lib_experimental_optional**

Variables

- constexpr in_place_t std::experimental::in_place
- constexpr nullopt_t std::experimental::nullopt

3.14.3.6.1 Detailed Description

Class template for optional values and surrounding facilities, as described in n3793 "A proposal to add a utility class to represent optional objects (Revision 5)".

3.14.3.6.2 Variable Documentation

in_place

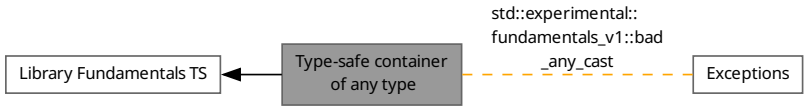
```
in_place_t std::experimental::fundamentals_v1::in_place  [constexpr]
```
Tag for in-place construction.

nullopt

```
nullopt_t std::experimental::fundamentals_v1::nullopt  [constexpr]
```
Tag to disengage optional objects.

3.14.3.7 Type-safe container of any type

Collaboration diagram for Type-safe container of any type:



Classes

- class std::experimental::fundamentals_v1::any
- class std::experimental::fundamentals_v1::bad_any_cast

Macros

- #define **__cpp_lib_experimental_any**

Functions

- static void **std::experimental::fundamentals_v1::any::_Manager_external< _Tp >::_S_manage** (_Op __↩ which, const any *__anyp, _Arg *__arg)
- static void **std::experimental::fundamentals_v1::any::_Manager_internal< _Tp >::_S_manage** (_Op __↩ which, const any *__anyp, _Arg *__arg)
- template<typename _ValueType>
  _ValueType std::experimental::any_cast (const any &__any)
- void std::experimental::swap (any &__x, any &__y) noexcept

- template<typename _ValueType>
  _ValueType std::experimental::any_cast (any &__any)
- template<typename _ValueType, typename enable_if<!is_move_constructible< _ValueType >::value||is_lvalue_reference< _ValueType >::value, bool >::type = true>
  _ValueType std::experimental::any_cast (any &&__any)

- template<typename _ValueType>
  const _ValueType * std::experimental::any_cast (const any *__any) noexcept
- template<typename _ValueType>
  _ValueType * std::experimental::any_cast (any *__any) noexcept

3.14.3.7.1 Detailed Description

A type-safe container for single values of value types, as described in n3804 "Any Library Proposal (Revision 3)".

3.14.3.7.2 Function Documentation

any_cast() [1/5]

```
template<typename _ValueType, typename enable_if<!is_move_constructible< _ValueType >::value||is_lvalue_reference
_ValueType >::value, bool >::type = true>
_ValueType std::experimental::fundamentals_v1::any_cast (
            any && __any)  [inline]
```

Access the contained object.

Template Parameters

_ValueType	A reference or CopyConstructible type.

Parameters

__any	The object to access.

Returns

The contained object.

Exceptions

bad_any_cast	If `__any.type() != typeid(remove_reference_t<_ValueType>)`

any_cast() [2/5]

```
template<typename _ValueType>
_ValueType std::experimental::fundamentals_v1::any_cast (
            any & __any)  [inline]
```

Access the contained object.

Template Parameters

_ValueType	A reference or CopyConstructible type.

Parameters

__any	The object to access.

Returns

The contained object.

Exceptions

bad_any_cast	If `__any.type() != typeid(remove_reference_t<_ValueType>)`

any_cast() [3/5]

```
template<typename _ValueType>
_ValueType * std::experimental::fundamentals_v1::any_cast (
            any * __any)  [inline], [noexcept]
```

Access the contained object.

Template Parameters

_ValueType	The type of the contained object.

Parameters

__any	A pointer to the object to access.

Returns

The address of the contained object if `__any != nullptr && __any.type() == typeid(_↩ ValueType)` , otherwise a null pointer.

any_cast() [4/5]

```
template<typename _ValueType>
_ValueType std::experimental::fundamentals_v1::any_cast (
            const any & __any)  [inline]
```

Access the contained object.

Template Parameters

_ValueType	A const-reference or CopyConstructible type.

Parameters

__any	The object to access.

Returns

The contained object.

Exceptions

bad_any_cast	If `__any.type() != typeid(remove_reference_t<_ValueType>)`

any_cast() [5/5]

```
template<typename _ValueType>
const _ValueType * std::experimental::fundamentals_v1::any_cast (
            const any * __any)  [inline], [noexcept]
```

Access the contained object.

Template Parameters

_ValueType	The type of the contained object.

Parameters

__any	A pointer to the object to access.

Returns

The address of the contained object if `__any != nullptr && __any.type() == typeid(_ValueType)` , otherwise a null pointer.

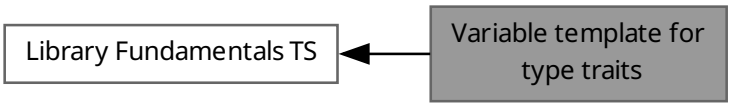
swap()

```
void std::experimental::fundamentals_v1::swap (
            any & __x,
            any & __y)  [inline], [noexcept]
```

Exchange the states of two `any` objects.

3.14.3.8 Variable template for type traits

Collaboration diagram for Variable template for type traits:



- #define **__cpp_lib_experimental_type_trait_variable_templates**
- template<typename _Tp>
  constexpr bool **std::experimental::is_void_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_null_pointer_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_integral_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_floating_point_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_array_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_pointer_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_lvalue_reference_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_rvalue_reference_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_member_object_pointer_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_member_function_pointer_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_enum_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_union_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_class_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_function_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_reference_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_arithmetic_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_fundamental_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_object_v**

- template<typename _Tp>
  constexpr bool **std::experimental::is_scalar_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_compound_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_member_pointer_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_const_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_volatile_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_trivial_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_trivially_copyable_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_standard_layout_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_pod_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_literal_type_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_empty_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_polymorphic_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_abstract_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_final_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_signed_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_unsigned_v**
- template<typename _Tp, typename... _Args>
  constexpr bool **std::experimental::is_constructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_default_constructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_copy_constructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_move_constructible_v**
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::is_assignable_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_copy_assignable_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_move_assignable_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_destructible_v**
- template<typename _Tp, typename... _Args>
  constexpr bool **std::experimental::is_trivially_constructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_trivially_default_constructible_v**

- template<typename _Tp>
  constexpr bool **std::experimental::is_trivially_copy_constructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_trivially_move_constructible_v**
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::is_trivially_assignable_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_trivially_copy_assignable_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_trivially_move_assignable_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_trivially_destructible_v**
- template<typename _Tp, typename... _Args>
  constexpr bool **std::experimental::is_nothrow_constructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_nothrow_default_constructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_nothrow_copy_constructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_nothrow_move_constructible_v**
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::is_nothrow_assignable_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_nothrow_copy_assignable_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_nothrow_move_assignable_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_nothrow_destructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::has_virtual_destructor_v**
- template<typename _Tp>
  constexpr size_t **std::experimental::alignment_of_v**
- template<typename _Tp>
  constexpr size_t **std::experimental::rank_v**
- template<typename _Tp, unsigned _Idx = 0>
  constexpr size_t **std::experimental::extent_v**
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::is_same_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_same_v< _Tp, _Tp >**
- template<typename _Base, typename _Derived>
  constexpr bool **std::experimental::is_base_of_v**
- template<typename _From, typename _To>
  constexpr bool **std::experimental::is_convertible_v**

3.14.3.8.1 Detailed Description

Since

Library Fundamentals TS v1. C++14.

See also

variable_templates

3.14.4 Parallelism TS

Collaboration diagram for Parallelism TS:

Topics

- Data parallel extensions

3.14.4.1 Detailed Description

Components defined by the *C++ Extensions for Parallelism* Technical Specification.

- ISO/IEC TS 19570:2015 C++ Extensions for Parallelism
- ISO/IEC TS 19570:2018 C++ Extensions for Parallelism, Version 2

3.14.4.2 Data parallel extensions

Collaboration diagram for Data parallel extensions:

Macros

- #define **__cpp_lib_experimental_parallel_simd**

3.14.4.2.1 Detailed Description

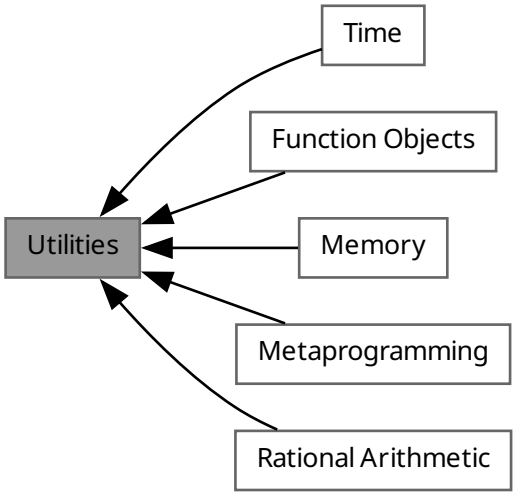
Data-parallel types library.

Since

C++17

3.15 Utilities

Collaboration diagram for Utilities:



Topics

- Function Objects
- Memory
- Metaprogramming
- Rational Arithmetic
- Time

Classes

- class std::bitset< _Nb >
- struct std::pair< _T1, _T2 >
- struct std::piecewise_construct_t
- class std::tuple< _Elements >
- class std::tuple< _T1, _T2 >
- struct std::tuple_element< __i, tuple< _Types... > >
- struct std::tuple_size< tuple< _Elements... > >
- struct std::type_index
- struct std::uses_allocator< tuple< _Types... >, _Alloc >

Typedefs

- template<typename _Tp, typename _Up>
  using **std::__like_t**

Functions

- template<typename... _Args1, typename... _Args2>
  constexpr std::pair< _T1, _T2 >::pair (piecewise_construct_t, tuple< _Args1... >, tuple< _Args2... >)
- template<typename _Tp>
  constexpr _Tp * std::__addressof (_Tp &__r) noexcept
- template<typename _Tp, typename _Up = _Tp>
  constexpr _Tp **std::__exchange** (_Tp &__obj, _Up &&__new_val)
- template<size_t __i, typename _Head, typename... _Tail>
  constexpr _Head & **std::__get_helper** (_Tuple_impl< __i, _Head, _Tail... > &__t) noexcept
- template<size_t __i, typename _Head, typename... _Tail>
  constexpr const _Head & **std::__get_helper** (const _Tuple_impl< __i, _Head, _Tail... > &__t) noexcept
- template<size_t __i, typename... _Types>
  __enable_if_t<(__i >=sizeof...(_Types))> **std::__get_helper** (const tuple< _Types... > &)=delete
- template<typename _Tp, typename _Up = typename __inv_unwrap<_Tp>::type>
  constexpr _Up && **std::__invfwd** (typename remove_reference< _Tp >::type &__t) noexcept
- template<typename _Callable, typename... _Args>
  constexpr __invoke_result< _Callable, _Args... >::type std::__invoke (_Callable &&__fn, _Args &&... __args) noexcept(__is_nothrow_invocable< _Callable, _Args... >::value)
- template<typename _Res, typename _MemFun, typename _Tp, typename... _Args>
  constexpr _Res **std::__invoke_impl** (__invoke_memfun_deref, _MemFun &&__f, _Tp &&__t, _Args &&... __↩
  args)
- template<typename _Res, typename _MemFun, typename _Tp, typename... _Args>
  constexpr _Res **std::__invoke_impl** (__invoke_memfun_ref, _MemFun &&__f, _Tp &&__t, _Args &&... __args)
- template<typename _Res, typename _MemPtr, typename _Tp>
  constexpr _Res **std::__invoke_impl** (__invoke_memobj_deref, _MemPtr &&__f, _Tp &&__t)
- template<typename _Res, typename _MemPtr, typename _Tp>
  constexpr _Res **std::__invoke_impl** (__invoke_memobj_ref, _MemPtr &&__f, _Tp &&__t)
- template<typename _Res, typename _Fn, typename... _Args>
  constexpr _Res **std::__invoke_impl** (__invoke_other, _Fn &&__f, _Args &&... __args)
- template<typename _Res, typename _Callable, typename... _Args>
  constexpr enable_if_t< is_invocable_r_v< _Res, _Callable, _Args... >, _Res > **std::__invoke_r** (_Callable &&__fn, _Args &&... __args) noexcept(is_nothrow_invocable_r_v< _Res, _Callable, _Args... >)
- template<typename _Cat, typename _Tp, typename _Up, typename _IndexSeq>
  constexpr _Cat **std::__tuple_cmp** (const _Tp &__t, const _Up &__u, _IndexSeq __indices)
- template<typename _Tp>
  constexpr _Tp * std::addressof (_Tp &__r) noexcept
- template<typename _Tp>
  const _Tp * **std::addressof** (const _Tp &&)=delete
- template<typename _Tp>
  auto std::declval () noexcept -> decltype(__declval< _Tp >(0))
- template<typename _Tp>
  constexpr _Tp && std::forward (typename std::remove_reference< _Tp >::type &&__t) noexcept
- template<typename _Tp>
  constexpr _Tp && std::forward (typename std::remove_reference< _Tp >::type &__t) noexcept
- template<typename... _Elements>
  constexpr tuple< _Elements &&... > std::forward_as_tuple (_Elements &&... __args) noexcept
- template<size_t __i, typename... _Elements>
  constexpr const __tuple_element_t< __i, tuple< _Elements... > > && std::get (const tuple< _Elements... > &&__t) noexcept
- template<size_t __i, typename... _Elements>
  constexpr const __tuple_element_t< __i, tuple< _Elements... > > & std::get (const tuple< _Elements... > &__t) noexcept

- template<size_t __i, typename... _Elements>
  constexpr __tuple_element_t< __i, tuple< _Elements... > > && std::get (tuple< _Elements... > &&__t) noexcept
- template<size_t __i, typename... _Elements>
  constexpr __tuple_element_t< __i, tuple< _Elements... > > & std::get (tuple< _Elements... > &__t) noexcept
- template<typename _T1, typename _T2>
  constexpr pair< typename __decay_and_strip< _T1 >::__type, typename __decay_and_strip< _T2 >::__type > std::make_pair (_T1 &&__x, _T2 &&__y)
- template<typename... _Elements>
  constexpr tuple< typename __decay_and_strip< _Elements >::__type... > std::make_tuple (_Elements &&... __args)
- template<typename _Tp>
  constexpr std::remove_reference< _Tp >::type && std::move (_Tp &&__t) noexcept
- template<typename _Tp>
  constexpr __conditional_t< __move_if_noexcept_cond< _Tp >::value, const _Tp &, _Tp && > std::move_if_noexcept (_Tp &__x) noexcept
- template<typename... _Tps, typename... _Ups>
  requires (sizeof...(_Tps) == sizeof...(_Ups)) && (requires { typename __detail::__synth3way_t<_Tps, _Ups>; } && ...)
  constexpr common_comparison_category_t< __detail::__synth3way_t< _Tps, _Ups >... > **std::operator<=>** (const tuple< _Tps... > &__t, const tuple< _Ups... > &__u)
- template<typename... _Tps, typename... _Ups>
  requires (sizeof...(_Tps) == sizeof...(_Ups)) && (requires (const _Tps& __t, const _Ups& __u) { { __t == __u } -> __detail::__boolean_testable; } && ...)
  constexpr bool **std::operator==** (const tuple< _Tps... > &__t, const tuple< _Ups... > &__u)
- template<typename _Tp>
  requires (! __is_tuple_like<_Tp>::value) && is_move_constructible_v<_Tp> && is_move_assignable_v<_Tp>
  constexpr void std::swap (_Tp &__a, _Tp &__b) noexcept(/*conditional */) is_nothrow_move_assignable< _Tp > >
- template<typename _Tp, size_t _Nm>
  requires is_swappable_v<_Tp>
  constexpr void std::swap (_Tp(&__a)[_Nm], _Tp(&__b)[_Nm]) noexcept(/*conditional */)
- template<typename... _Elements>
  constexpr enable_if< __and_< __is_swappable< _Elements >... >::value >::type std::swap (tuple< _Elements... > &__x, tuple< _Elements... > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename... _Elements>
  constexpr tuple< _Elements &... > std::tie (_Elements &... __args) noexcept
- template<typename... _UTypes>
  std::tuple (_UTypes...) -> tuple< _UTypes... >
- template<typename _Alloc, typename... _UTypes>
  std::tuple (allocator_arg_t, _Alloc, _UTypes...) -> tuple< _UTypes... >
- template<typename _Alloc, typename _T1, typename _T2>
  std::tuple (allocator_arg_t, _Alloc, pair< _T1, _T2 >) -> tuple< _T1, _T2 >
- template<typename _Alloc, typename... _UTypes>
  std::tuple (allocator_arg_t, _Alloc, tuple< _UTypes... >) -> tuple< _UTypes... >
- template<typename _T1, typename _T2>
  std::tuple (pair< _T1, _T2 >) -> tuple< _T1, _T2 >
- template<typename... _Tpls, typename = typename enable_if<__and_<__is_tuple_like<_Tpls>...>::value>::type>
  constexpr auto std::tuple_cat (_Tpls &&... __tpls) -> typename __tuple_cat_result< _Tpls... >::__type

Variables

- constexpr piecewise_construct_t std::piecewise_construct

- template<typename... _Types>
  constexpr size_t **std::tuple_size_v**< **const tuple**< **_Types...** > >
- template<typename... _Types>
  constexpr size_t **std::tuple_size_v**< **tuple**< **_Types...** > >

- template<typename _T1, typename _T2>
  pair (_T1, _T2) -> pair< _T1, _T2 >
- template<typename _T1, typename _T2, typename _U1, typename _U2>
  constexpr bool operator== (const pair< _T1, _T2 > &__x, const pair< _U1, _U2 > &__y)
- template<typename _T1, typename _T2, typename _U1, typename _U2>
  constexpr common_comparison_category_t< __detail::__synth3way_t< _T1, _U1 >, __detail::__synth3way_t< _T2, _U2 > > operator<=> (const pair< _T1, _T2 > &__x, const pair< _U1, _U2 > &__y)
- template<typename _T1, typename _T2>
  constexpr enable_if< __and_< __is_swappable< _T1 >, __is_swappable< _T2 > >::value >::type swap (pair< _T1, _T2 > &__x, pair< _T1, _T2 > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename _T1, typename _T2>
  constexpr enable_if< __and_< __is_swappable< _T1 >, __is_swappable< _T2 > >::value >::type swap (pair< _T1, _T2 > &__x, pair< _T1, _T2 > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename _T1, typename _T2, typename _U1, typename _U2>
  constexpr bool operator== (const pair< _T1, _T2 > &__x, const pair< _U1, _U2 > &__y)
- template<typename _T1, typename _T2, typename _U1, typename _U2>
  constexpr common_comparison_category_t< __detail::__synth3way_t< _T1, _U1 >, __detail::__synth3way_t< _T2, _U2 > > operator<=> (const pair< _T1, _T2 > &__x, const pair< _U1, _U2 > &__y)

3.15.1 Detailed Description

Basic function and class templates used with the rest of the library. Includes pair, swap, forward/move helpers, declval, integer_sequence.

3.15.2 Function Documentation

pair() [1/2]

```
template<typename _T1, typename _T2>
pair (
            _T1 ,
            _T2 ) -> pair< _T1, _T2 >  [related]
```

Two pairs are equal iff their members are equal.

pair() [2/2]

```
template<class _T1, class _T2>
template<typename...  _Args1, typename...  _Args2>
std::pair< _T1, _T2 >::pair (
            piecewise_construct_t ,
            tuple< _Args1...  > __first,
            tuple< _Args2...  > __second)  [inline], [constexpr]
```

"piecewise construction" using a tuple of arguments for each member.

Parameters

__first	Arguments for the first member of the pair.
__second	Arguments for the second member of the pair.

The elements of each tuple will be used as the constructor arguments for the data members of the pair.

__addressof()

```
template<typename _Tp>
_Tp * std::__addressof (
            _Tp & __r)  [inline], [constexpr], [noexcept]
```

Same as C++11 std::addressof.

Referenced by std::back_insert_iterator< _Container >::back_insert_iterator(), std::front_insert_iterator< _Container >::front_insert_iter
std::insert_iterator< _Container >::insert_iterator(), std::istream_iterator< _Tp, _CharT, _Traits, _Dist >::istream_iterator(),
std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator(), std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator(),
__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if(), std::basic_ios< _CharT, _Traits >::copyfmt(),
__gnu_debug::basic_string< char >::find_last_of(), std::forward_list< _Tp, _Alloc >::merge(), std::list< _Tp, _Alloc >::merge(),
std::list< _Tp, _Alloc >::merge(), std::sub_match< const char * >::operator<<(), __gnu_debug::_Safe_iterator< _Base_iterator, map >
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=(), std::deque< _Tp, _Alloc >::operator=(),
std::forward_list< _Tp, _Alloc >::operator=(), std::list< _Tp, _Alloc >::operator=(), std::vector< _Tp, _Alloc >::operator=(),
std::sub_match< const char * >::operator==(), std::vector< _Tp, polymorphic_allocator< _Tp > >< _BigBlock >::reserve(),
std::list< _Tp, polymorphic_allocator< _Tp > >::splice(), and std::list< _Tp, polymorphic_allocator< _Tp > >::splice().

__invoke()

```
template<typename _Callable, typename...  _Args>
__invoke_result< _Callable, _Args...  >::type std::__invoke (
            _Callable && __fn,
            _Args &&...  __args)  [constexpr], [noexcept]
```

Invoke a callable object.

References forward().

addressof()

```
template<typename _Tp>
_Tp * std::addressof (
            _Tp & __r)  [inline], [constexpr], [noexcept]
```

Returns the actual address of the object or function referenced by r, even in the presence of an overloaded operator&.

Parameters

↩ _↩ ↩ _↩ *r*	Reference to an object or function.

Returns

The actual address.

Since

C++11

Referenced by std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::function_ref(), std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::function_ref(), _Destroy(), _Destroy_n(), std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::operator=(), and std::move_only_function< _Res

declval()

```
template<typename _Tp>
auto std::declval () -> decltype(__declval< _Tp >(0))  [noexcept]
```

Utility to simplify expressions used in unevaluated operands

Since

C++11

Referenced by std::vector< _Tp, polymorphic_allocator< _Tp > >::vector(), and std::unique_ptr< _State >::operator*().

forward() [1/2]

```
template<typename _Tp>
_Tp && std::forward (
            typename std::remove_reference< _Tp >::type && __t)  [constexpr], [noexcept]
```

Forward an rvalue.

Returns

The parameter cast to the specified type.

This function is used to implement "perfect forwarding".

Since

C++11

forward() [2/2]

```
template<typename _Tp>
_Tp && std::forward (
            typename std::remove_reference< _Tp >::type & __t)  [constexpr], [noexcept]
```

Forward an lvalue.

Returns

The parameter cast to the specified type.

This function is used to implement "perfect forwarding".

Since

C++11

Referenced by std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::copyable_function(), std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::copyable_function(), std::copyable_function< _R std::function< _Res(_ArgTypes...)>::function(), std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::move_only_function(), std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::move_only_function(), std::unique_ptr< _State >::unique_ptr(), __invoke(), _Construct(), std::list< _Tp, polymorphic_allocator< _Tp > >::_M_create_node(), std::deque< _Tp, _Alloc >::_M_push_back_aux(), std::deque< _Tp, _Alloc >::_M_push_front_aux(), std::shared_ptr< _State_base >::a std::basic_string< _CharT >::append(), std::allocator_traits< _OuterAlloc >::construct(), std::allocator_traits< allocator< _Tp > >::cons std::allocator_traits< allocator< void > >::construct(), std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::construct(), std::deque< _Tp, _Alloc >::emplace(), std::list< _Tp, _Alloc >::emplace(), std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< c std::multimap< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::emplace(), std::multiset< _Key, _Cmp, polymo std::set< _Key, _Cmp, polymorphic_allocator< _Key > >::emplace(), std::unordered_map< _Key, _Tp, _Hash, _Pred, polymorphic_allo std::unordered_multimap< _Key, _Tp, _Hash, _Pred, polymorphic_allocator< pair< const _Key, _Tp > > >::emplace(), std::unordered_multiset< _Key, _Hash, _Pred, polymorphic_allocator< _Key > >::emplace(), std::unordered_set< _Key, _Hash, _Pred,

std::vector< _Tp, polymorphic_allocator< _Tp > >::emplace(), std::forward_list< _Tp, polymorphic_allocator< _Tp > >::emplace_after
std::forward_list< _Tp, polymorphic_allocator< _Tp > >::emplace_front(), std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< c
std::multimap< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::emplace_hint(), std::multiset< _Key, _Cmp, po
std::set< _Key, _Cmp, polymorphic_allocator< _Key > >::emplace_hint(), std::unordered_map< _Key, _Tp, _Hash, _Pred, polymorphic
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, polymorphic_allocator< pair< const _Key, _Tp > > >::emplace_hint(),
std::unordered_multiset< _Key, _Hash, _Pred, polymorphic_allocator< _Key > >::emplace_hint(), std::unordered_set< _Key, _Hash, _
get(), get(), get(), get(), std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert(),
std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert(), std::multimap< _Key, _Tp, _Cmp, polymor
std::multimap< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert(), std::unordered_map< _Key, _Tp, _Ha
std::unordered_map< _Key, _Tp, _Hash, _Pred, polymorphic_allocator< pair< const _Key, _Tp > > >::insert(),
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, polymorphic_allocator< pair< const _Key, _Tp > > >::insert(),
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, polymorphic_allocator< pair< const _Key, _Tp > > >::insert(),
std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert_or_assign(), std::map< _Key, _Tp, _Cmp, po
make_pair(), std::shared_ptr< _State_base >::make_shared(), std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexc
std::function< _Res(_ArgTypes...)>::operator()(), std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::opera
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::operator()(), std::copyable_function< _Res(_Ar
std::function< _Res(_ArgTypes...)>::operator=(), std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)
std::unique_ptr< _State >::operator=(), std::unique_ptr< _Tp[], _Dp >::operator=(), sample(), and throw_with_nested().

forward_as_tuple()

```
template<typename... _Elements>
tuple< _Elements &&... > std::forward_as_tuple (
            _Elements &&... __args) [constexpr], [noexcept]
```

Create a tuple of lvalue or rvalue references to the arguments.
Referenced by std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert_or_assign(), and std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert_or_assign().

get() [1/4]

```
template<size_t __i, typename... _Elements>
const __tuple_element_t< __i, tuple< _Elements... > > && std::get (
            const tuple< _Elements... > && __t) [constexpr], [noexcept]
```

Return a const rvalue reference to the ith element of a const tuple rvalue.

get() [2/4]

```
template<size_t __i, typename... _Elements>
const __tuple_element_t< __i, tuple< _Elements... > > & std::get (
            const tuple< _Elements... > & __t) [constexpr], [noexcept]
```

Return a const reference to the ith element of a const tuple.

get() [3/4]

```
template<size_t __i, typename... _Elements>
__tuple_element_t< __i, tuple< _Elements... > > && std::get (
            tuple< _Elements... > && __t) [constexpr], [noexcept]
```

Return an rvalue reference to the ith element of a tuple rvalue.

get() [4/4]

```
template<size_t __i, typename... _Elements>
__tuple_element_t< __i, tuple< _Elements... > > & std::get (
            tuple< _Elements... > & __t) [constexpr], [noexcept]
```

Return a reference to the ith element of a tuple.

make_pair()

```
template<typename _T1, typename _T2>
pair< typename __decay_and_strip< _T1 >::__type, typename __decay_and_strip< _T2 >::__type >
std::make_pair (
            _T1 && __x,
            _T2 && __y)  [constexpr]
```

A convenience wrapper for creating a pair from two objects.

Parameters

_↩ _x	The first object.
_↩ _y	The second object.

Returns

A newly-constructed pair<> object of the appropriate type.

The C++98 standard says the objects are passed by reference-to-const, but C++03 says they are passed by value (this was LWG issue #181).
Since C++11 they have been passed by forwarding reference and then forwarded to the new members of the pair. To create a pair with a member of reference type, pass a `reference_wrapper` to this function.
References forward().
Referenced by __gnu_parallel::__find_template(), __gen_two_uniform_ints(), __gnu_debug::__get_distance(), __gnu_parallel::__parallel_merge_advance(), __gnu_parallel::__parallel_sort_qsb(), __gnu_parallel::__qsb_local_sort_with_helping(), __gnu_parallel::__adjacent_find_selector::_M_sequential_algorithm(), __gnu_parallel::__find_first_of_selector< _FIterator >::_M_seque __gnu_parallel::__find_if_selector::_M_sequential_algorithm(), __gnu_parallel::multiseq_partition(), __gnu_parallel::multiseq_selection(), __gnu_parallel::parallel_multiway_merge(), __gnu_parallel::parallel_sort_mwms_pu(), and __gnu_pbds::detail::pat_trie_base::_Node_cite

make_tuple()

```
template<typename... _Elements>
tuple< typename __decay_and_strip< _Elements >::__type... > std::make_tuple (
            _Elements &&... __args)  [constexpr]
```

Create a tuple containing copies of the arguments.

move()

```
template<typename _Tp>
std::remove_reference< _Tp >::type && std::move (
            _Tp && __t)  [constexpr], [noexcept]
```

Convert a value to an rvalue.

Parameters

↩ _↩ ↩ _↩ t	A thing of arbitrary type.

Returns

The parameter cast to an rvalue-reference to allow moving it.

Since

C++11

Referenced by __gnu_cxx::__versa_string< char >::__versa_string(), std::basic_string< _CharT >::basic_string(), std::deque< _Tp, polymorphic_allocator< _Tp > >::deque(), std::discard_block_engine< ranlux24_base, 223, 23 >::discard_block_eng std::forward_list< _Tp, polymorphic_allocator< _Tp > >::forward_list(), std::independent_bits_engine< _RandomNumberEngine, __w, _ std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::map(), std::match_results< _BidirectionalIterator, p std::multimap< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::multimap(), std::multiset< _Key, _Cmp, polymo std::set< _Key, _Cmp, polymorphic_allocator< _Key > >::set(), std::shared_ptr< _State_base >::shared_ptr(), std::shared_ptr< _State_base >::shared_ptr(), std::shared_ptr< _State_base >::shared_ptr(), std::shared_ptr< _State_base >::shared std::shared_ptr< _State_base >::shared_ptr(), std::shared_ptr< _State_base >::shared_ptr(), std::shared_ptr< _State_base >::shared std::shuffle_order_engine< minstd_rand0, 256 >::shuffle_order_engine(), std::unique_ptr< _State >::unique_ptr(), std::unique_ptr< _Tp[ ], _Dp >::unique_ptr(), std::vector< _Tp, polymorphic_allocator< _Tp > >::vector(), std::unique_ptr< _State >::~ std::basic_regex< char >::assign(), std::basic_string< _CharT >::assign(), atomic_compare_exchange_strong(), atomic_compare_exchange_strong(), atomic_compare_exchange_strong_explicit(), atomic_compare_exchange_strong_explicit(), atomic_compare_exchange_weak(), atomic_compare_exchange_weak(), atomic_compare_exchange_weak_explicit(), atomic_compare_exchange_weak_explicit(), atomic_exchange(), atomic_exchange(), atomic_store(), atomic_store(), std::shared_ptr< _State_base >::const_pointer_cast(), copy_n(), distance(), std::shared_ptr< _State_base >::dynamic_pointer_cast(), for_each_n(), get(), get(), std::deque< _Tp, polymorphic_allocator< _Tp > >::insert(), std::list< _Tp, polymorphic_allocator< _Tp > >:: std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert(), std::map< _Key, _Tp, _Cmp, polymorphic_ std::multimap< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert(), std::multimap< _Key, _Tp, _Cmp, poly std::unordered_map< _Key, _Tp, _Hash, _Pred, polymorphic_allocator< pair< const _Key, _Tp > > >::insert(), std::unordered_map< _Key, _Tp, _Hash, _Pred, polymorphic_allocator< pair< const _Key, _Tp > > >::insert(), std::unordered_multimap< _Key, _Tp, _Hash, _Pred, polymorphic_allocator< pair< const _Key, _Tp > > >::insert(), std::unordered_multimap< _Key, _Tp, _Hash, _Pred, polymorphic_allocator< pair< const _Key, _Tp > > >::insert(), std::unordered_multiset< _Key, _Hash, _Pred, polymorphic_allocator< _Key > >::insert(), std::unordered_multiset< _Key, _Hash, _Pre std::unordered_set< _Key, _Hash, _Pred, polymorphic_allocator< _Key > >::insert(), std::unordered_set< _Key, _Hash, _Pred, polymo std::vector< _Tp, _Alloc >::insert(), std::vector< _Tp, polymorphic_allocator< _Tp > >::insert(), std::forward_list< _Tp, _Alloc >::merge std::forward_list< _Tp, polymorphic_allocator< _Tp > >::merge(), std::forward_list< _Tp, polymorphic_allocator< _Tp > >::merge(), std::list< _Tp, polymorphic_allocator< _Tp > >::merge(), std::list< _Tp, polymorphic_allocator< _Tp > >::merge(), move_if_noexcept(), std::chrono::operator<<(), std::deque< _Tp, polymorphic_allocator< _Tp > >::operator=(), std::forward_list< _Tp, polymorphic_allocator< _Tp > >::operator=(), std::function< _Res(_ArgTypes...)>::operator=(), std::list< _Tp, polymorphic_allocator< _Tp > >::operator=(), std::vector< _Tp, polymorphic_allocator< _Tp > >::operator=(), std::filesystem::path::operator>>, std::unordered_map< _Key, _Tp, _Hash, _Pred, polymorphic_allocator< pair< const _Key, _Tp > > std::shared_ptr< _State_base >::reinterpret_pointer_cast(), std::unique_ptr< _State >::reset(), std::unique_ptr< _Tp[ ], _Dp >::reset(), std::list< _Tp, polymorphic_allocator< _Tp > >::splice(), std::list< _Tp, polymorphic_allocator< _Tp > >::splice(), std::forward_list< _Tp, polymorphic_allocator< _Tp > >::splice_after(), and std::shared_ptr< _State_base >::static_pointer_cast().

move_if_noexcept()

```
template<typename _Tp>
__conditional_t< __move_if_noexcept_cond< _Tp >::value, const _Tp &, _Tp && > std::move_if_↩
noexcept (
            _Tp & __x)  [constexpr], [noexcept]
```

Conditionally convert a value to an rvalue.

Parameters

_↩ _x	A thing of arbitrary type.

Returns

The parameter, possibly cast to an rvalue-reference.

Same as std::move unless the type's move constructor could throw and the type is copyable, in which case an lvalue-reference is returned instead.

Since

C++11

References move().

operator<=>() [1/2]

```
template<typename _T1, typename _T2, typename _U1, typename _U2>
common_comparison_category_t< __detail::__synth3way_t< _T1, _U1 >, __detail::__synth3way_t< _T2,
_U2 > > operator<=> (
            const pair< _T1, _T2 > & __x,
            const pair< _U1, _U2 > & __y)  [related]
```

Defines a lexicographical order for pairs.
For two pairs of comparable types, `P` is ordered before `Q` if `P.first` is less than `Q.first`, or if `P.first` and `Q.first` are equivalent (neither is less than the other) and `P.second` is less than `Q.second`.

operator<=>() [2/2]

```
template<typename _T1, typename _T2, typename _U1, typename _U2>
common_comparison_category_t< __detail::__synth3way_t< _T1, _U1 >, __detail::__synth3way_t< _T2,
_U2 > > operator<=> (
            const pair< _T1, _T2 > & __x,
            const pair< _U1, _U2 > & __y)  [related]
```

Defines a lexicographical order for pairs.
For two pairs of comparable types, `P` is ordered before `Q` if `P.first` is less than `Q.first`, or if `P.first` and `Q.first` are equivalent (neither is less than the other) and `P.second` is less than `Q.second`.

operator==() [1/2]

```
template<typename _T1, typename _T2, typename _U1, typename _U2>
bool operator== (
            const pair< _T1, _T2 > & __x,
            const pair< _U1, _U2 > & __y)  [related]
```

Two pairs are equal iff their members are equal.

operator==() [2/2]

```
template<typename _T1, typename _T2, typename _U1, typename _U2>
bool operator== (
            const pair< _T1, _T2 > & __x,
            const pair< _U1, _U2 > & __y)  [related]
```

Two pairs are equal iff their members are equal.

swap() [1/5]

```
template<typename _T1, typename _T2>
enable_if< __and_< __is_swappable< _T1 >, __is_swappable< _T2 > >::value >::type swap (
            pair< _T1, _T2 > & __x,
            pair< _T1, _T2 > & __y)  [related]
```

Swap overload for pairs. Calls std::pair::swap().

Note

This std::swap overload is not declared in C++03 mode, which has performance implications, e.g. see https://gcc.gnu.org/PR38466

swap() [2/5]

```
template<typename _T1, typename _T2>
enable_if< __and_< __is_swappable< _T1 >, __is_swappable< _T2 > >::value >::type swap (
            pair< _T1, _T2 > & __x,
            pair< _T1, _T2 > & __y)  [related]
```

Swap overload for pairs. Calls std::pair::swap().

Note

This std::swap overload is not declared in C++03 mode, which has performance implications, e.g. see https://gcc.gnu.org/PR38466

swap() [3/5]

```
template<typename _Tp>
requires (!  __is_tuple_like<_Tp>::value) && is_move_constructible_v<_Tp> && is_move_assignable_v<_Tp>
void std::swap (
            _Tp & __a,
            _Tp & __b)  [inline], [constexpr], [noexcept]
```

Swaps two values.

Parameters

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

Returns

Nothing.

swap() [4/5]

```
template<typename _Tp, size_t _Nm>
requires is_swappable_v<_Tp>
void std::swap (
            _Tp(&) __a[_Nm],
            _Tp(&) __b[_Nm])  [inline], [constexpr], [noexcept]
```

Swap the contents of two arrays.

swap() [5/5]

```
template<typename...  _Elements>
enable_if< __and_< __is_swappable< _Elements >...  >::value >::type std::swap (
            tuple< _Elements...  > & __x,
            tuple< _Elements...  > & __y)  [inline], [constexpr], [delete], [noexcept]
```

Exchange the values of two tuples.

tie()

```
template<typename... _Elements>
tuple< _Elements &... > std::tie (
            _Elements &... __args) [constexpr], [noexcept]
```

Return a tuple of lvalue references bound to the arguments.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt().

tuple_cat()

```
template<typename... _Tpls, typename = typename enable_if<__and_<__is_tuple_like<_Tpls>...>↩
::value>::type>
auto std::tuple_cat (
            _Tpls &&... __tpls) -> typename __tuple_cat_result<_Tpls...>::__type [constexpr]
```

Create a `tuple` containing all elements from multiple tuple-like objects.

3.15.3 Variable Documentation

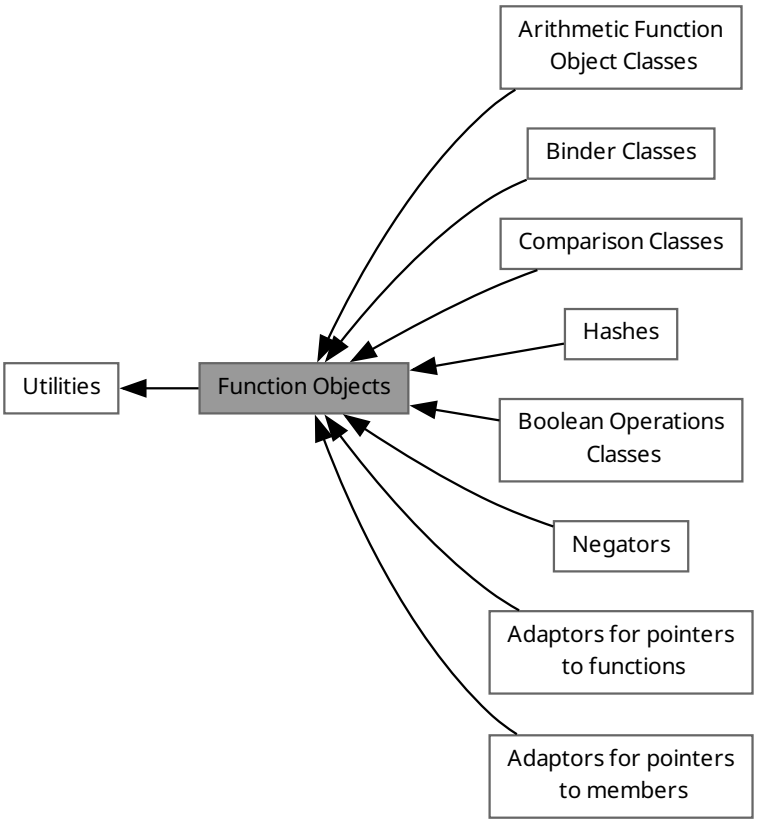
piecewise_construct

```
piecewise_construct_t std::piecewise_construct [inline], [constexpr]
```

Tag for piecewise construction of std::pair objects.
Referenced by std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert_or_assign(), std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert_or_assign(), and std::map< _Key, _Tp, _Cm

3.15.4 Function Objects

Collaboration diagram for Function Objects:



Topics

- Adaptors for pointers to functions
- Adaptors for pointers to members
- Arithmetic Function Object Classes
- Binder Classes
- Boolean Operations Classes
- Comparison Classes
- Hashes
- Negators

Classes

- struct std::binary_function< _Arg1, _Arg2, _Result >
- class std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>
- class std::function< _Res(_ArgTypes...)>
- class std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>

- class std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>
- class std::reference_wrapper< _Tp >
- struct std::unary_function< _Arg, _Result >

Functions

- template<typename _Tp, typename _Class>
  constexpr _Mem_fn< _Tp _Class::* > std::mem_fn (_Tp _Class::*__pm) noexcept

3.15.4.1 Detailed Description

Function objects, or *functors*, are objects with an `operator()` defined and accessible. They can be passed as arguments to algorithm templates and used in place of a function pointer. Not only is the resulting expressiveness of the library increased, but the generated code can be more efficient than what you might write by hand. When we refer to *functors*, then, generally we include function pointers in the description as well.
Often, functors are only created as temporaries passed to algorithm calls, rather than being created as named variables. Two examples taken from the standard itself follow. To perform a by-element addition of two vectors `a` and `b` containing `double`, and put the result in `a`, use

```
transform (a.begin(), a.end(), b.begin(), a.begin(), plus<double>());
```

To negate every element in `a`, use

```
transform(a.begin(), a.end(), a.begin(), negate<double>());
```

The addition and negation functions will usually be inlined directly.
An *adaptable function object* is one which provides nested typedefs `result_type` and either `argument_type` (for a unary function) or `first_argument_type` and `second_argument_type` (for a binary function). Those typedefs are used by function object adaptors such as `bind2nd`. The standard library provides two class templates, `unary_function` and `binary_function`, which define those typedefs and so can be used as base classes of adaptable function objects.
Since C++11 the use of function object adaptors has been superseded by more powerful tools such as lambda expressions, `function<>`, and more powerful type deduction (using `auto` and `decltype`). The helpers for defining adaptable function objects are deprecated since C++11, and no longer part of the standard library since C++17. However, they are still defined and used by libstdc++ after C++17, as a conforming extension.

3.15.4.2 Function Documentation

mem_fn()

```
template<typename _Tp, typename _Class>
_Mem_fn< _Tp _Class::* > std::mem_fn (
            _Tp _Class::* __pm)  [inline], [constexpr], [noexcept]
```

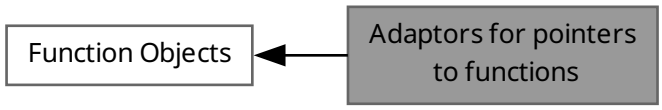
Returns a function object that forwards to the member pointer pointer `pm`.
This allows a pointer-to-member to be transformed into a function object that can be called with an object expression as its first argument.
For a pointer-to-data-member the result must be called with exactly one argument, the object expression that would be used as the first operand in a `obj.*memptr` or `objp->*memptr` expression.
For a pointer-to-member-function the result must be called with an object expression and any additional arguments to pass to the member function, as in an expression like `(obj.*memfun)(args...)` or `(objp->*memfun)(args...)`.
The object expression can be a pointer, reference, `reference_wrapper`, or smart pointer, and the call wrapper will dereference it as needed to apply the pointer-to-member.

Since

C++11

3.15.4.3 Adaptors for pointers to functions

Collaboration diagram for Adaptors for pointers to functions:



Classes

- class std::pointer_to_binary_function< _Arg1, _Arg2, _Result >
- class std::pointer_to_unary_function< _Arg, _Result >

Functions

- template<typename _Arg, typename _Result>
  pointer_to_unary_function< _Arg, _Result > std::ptr_fun (_Result(*__x)(_Arg))
- template<typename _Arg1, typename _Arg2, typename _Result>
  pointer_to_binary_function< _Arg1, _Arg2, _Result > std::ptr_fun (_Result(*__x)(_Arg1, _Arg2))

3.15.4.3.1 Detailed Description

The advantage of function objects over pointers to functions is that the objects in the standard library declare nested typedefs describing their argument and result types with uniform names (e.g., `result_type` from the base classes `unary_function` and `binary_function`). Sometimes those typedefs are required, not just optional.
Adaptors are provided to turn pointers to unary (single-argument) and binary (double-argument) functions into function objects. The long-winded functor `pointer_to_unary_function` is constructed with a function pointer `f`, and its `operator()` called with argument `x` returns `f(x)`. The functor `pointer_to_binary_function` does the same thing, but with a double-argument `f` and `operator()`.
The function `ptr_fun` takes a pointer-to-function `f` and constructs an instance of the appropriate functor.

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

3.15.4.3.2 Function Documentation

ptr_fun() [1/2]

```
template<typename _Arg, typename _Result>
pointer_to_unary_function< _Arg, _Result > std::ptr_fun (
            _Result(* __x )(_Arg))  [inline]
```

One of the adaptors for function pointers.
References ptr_fun().
Referenced by ptr_fun(), and ptr_fun().

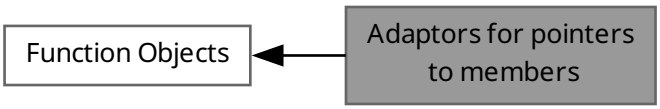
ptr_fun() [2/2]

```
template<typename _Arg1, typename _Arg2, typename _Result>
pointer_to_binary_function< _Arg1, _Arg2, _Result > std::ptr_fun (
            _Result(* __x )(_Arg1, _Arg2))  [inline]
```

One of the adaptors for function pointers.
References ptr_fun().

3.15.4.4 Adaptors for pointers to members

Collaboration diagram for Adaptors for pointers to members:



Classes

- class std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >
- class std::const_mem_fun1_t< _Ret, _Tp, _Arg >
- class std::const_mem_fun_ref_t< _Ret, _Tp >
- class std::const_mem_fun_t< _Ret, _Tp >
- class std::mem_fun1_ref_t< _Ret, _Tp, _Arg >
- class std::mem_fun1_t< _Ret, _Tp, _Arg >
- class std::mem_fun_ref_t< _Ret, _Tp >
- class std::mem_fun_t< _Ret, _Tp >

Functions

- template<typename _Ret, typename _Tp>
  mem_fun_t< _Ret, _Tp > **std::mem_fun** (_Ret(_Tp::*__f)())
- template<typename _Ret, typename _Tp, typename _Arg>
  mem_fun1_t< _Ret, _Tp, _Arg > **std::mem_fun** (_Ret(_Tp::*__f)(_Arg))
- template<typename _Ret, typename _Tp>
  mem_fun_ref_t< _Ret, _Tp > **std::mem_fun_ref** (_Ret(_Tp::*__f)())
- template<typename _Ret, typename _Tp, typename _Arg>
  mem_fun1_ref_t< _Ret, _Tp, _Arg > **std::mem_fun_ref** (_Ret(_Tp::*__f)(_Arg))

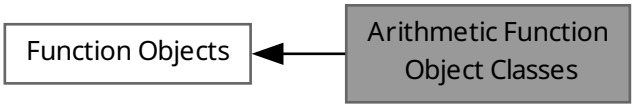
3.15.4.4.1 Detailed Description

There are a total of 8 = 2^3 function objects in this family. (1) Member functions taking no arguments vs member functions taking one argument. (2) Call through pointer vs call through reference. (3) Const vs non-const member function.
All of this complexity is in the function objects themselves. You can ignore it by using the helper function `mem_fun` and `mem_fun_ref`, which create whichever type of adaptor is appropriate.

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

3.15.4.5 Arithmetic Function Object Classes

Collaboration diagram for Arithmetic Function Object Classes:



Classes

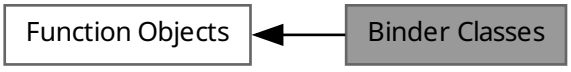
- struct std::divides< _Tp >
- struct std::divides< void >
- struct std::minus< _Tp >
- struct std::minus< void >
- struct std::modulus< _Tp >
- struct std::modulus< void >
- struct std::multiplies< _Tp >
- struct std::multiplies< void >
- struct std::negate< _Tp >
- struct std::negate< void >
- struct std::plus< _Tp >

3.15.4.5.1 Detailed Description

The library provides function objects for basic arithmetic operations. See the documentation for function objects for examples of their use.

3.15.4.6 Binder Classes

Collaboration diagram for Binder Classes:



Namespaces

- namespace std::placeholders

Classes

- struct std::_Placeholder< _Num >
- class std::binder1st< _Operation >
- class std::binder2nd< _Operation >
- struct std::is_bind_expression< _Tp >
- struct std::is_bind_expression< _Bind< _Signature > >
- struct std::is_bind_expression< _Bind_result< _Result, _Signature > >
- struct std::is_bind_expression< const _Bind< _Signature > >
- struct std::is_bind_expression< const _Bind_result< _Result, _Signature > >
- struct std::is_bind_expression< const volatile _Bind< _Signature > >
- struct std::is_bind_expression< const volatile _Bind_result< _Result, _Signature > >
- struct std::is_bind_expression< volatile _Bind< _Signature > >
- struct std::is_bind_expression< volatile _Bind_result< _Result, _Signature > >
- struct std::is_placeholder< _Tp >
- struct std::is_placeholder< _Placeholder< _Num > >

Functions

- template<typename _Func, typename... _BoundArgs>
  constexpr _Bind_helper< __is_socketlike< _Func >::value, _Func, _BoundArgs... >::type std::bind (_Func &&↩ __f, _BoundArgs &&... __args)
- template<typename _Result, typename _Func, typename... _BoundArgs>
  constexpr _Bindres_helper< _Result, _Func, _BoundArgs... >::type std::bind (_Func &&__f, _BoundArgs &&... __args)
- template<typename _Operation, typename _Tp>
  binder1st< _Operation > std::bind1st (const _Operation &__fn, const _Tp &__x)
- template<typename _Operation, typename _Tp>
  binder2nd< _Operation > std::bind2nd (const _Operation &__fn, const _Tp &__x)

3.15.4.6.1 Detailed Description

Binders turn functions/functors with two arguments into functors with a single argument, storing an argument to be applied later. For example, a variable `B` of type `binder1st` is constructed from a functor `f` and an argument `x`. Later, B's `operator()` is called with a single argument `y`. The return value is the value of `f(x,y)`. `B` can be *called* with various arguments (y1, y2, ...) and will in turn call `f(x,y1)`, `f(x,y2)`, ...

The function `bind1st` is provided to save some typing. It takes the function and an argument as parameters, and returns an instance of `binder1st`.

The type `binder2nd` and its creator function `bind2nd` do the same thing, but the stored argument is passed as the second parameter instead of the first, e.g., `bind2nd`(std::minus<float>(),1.3) will create a functor whose `operator()` accepts a floating-point number, subtracts 1.3 from it, and returns the result. (If `bind1st` had been used, the functor would perform *1.3 - x* instead.

Creator-wrapper functions like `bind1st` are intended to be used in calling algorithms. Their return values will be temporary objects. (The goal is to not require you to type names like `std::binder1st<std::plus<int>>` for declaring a variable to hold the return value from `bind1st(std::plus<int>(),5)`.

These become more useful when combined with the composition functions.

These functions are deprecated in C++11 and can be replaced by `std::bind` (or `std::tr1::bind`) which is more powerful and flexible, supporting functions with any number of arguments. Uses of `bind1st` can be replaced by `std::bind(f, x, std::placeholders::_1)` and `bind2nd` by `std::bind(f, std↩ ::placeholders::_1, x)`.

3.15.4.6.2 Function Documentation

bind() [1/2]

```
template<typename _Func, typename...  _BoundArgs>
_Bind_helper< __is_socketlike< _Func >::value, _Func, _BoundArgs...  >::type std::bind (
            _Func && __f,
            _BoundArgs &&...  __args)  [inline], [constexpr]
```

Function template for std::bind.

Since

C++11

bind() [2/2]

```
template<typename _Result, typename _Func, typename...  _BoundArgs>
_Bindres_helper< _Result, _Func, _BoundArgs...  >::type std::bind (
            _Func && __f,
            _BoundArgs &&...  __args)  [inline], [constexpr]
```

Function template for std::bind<R>.

Since

C++11

bind1st()

```
template<typename _Operation, typename _Tp>
binder1st< _Operation > std::bind1st (
            const _Operation & __fn,
            const _Tp & __x)  [inline]
```

One of the binder functors.
References bind1st().
Referenced by bind1st().

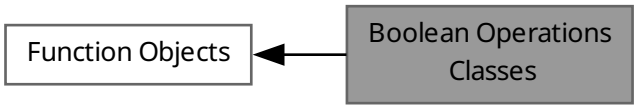
bind2nd()

```
template<typename _Operation, typename _Tp>
binder2nd< _Operation > std::bind2nd (
            const _Operation & __fn,
            const _Tp & __x)  [inline]
```

One of the binder functors.
References bind2nd().
Referenced by bind2nd().

3.15.4.7 Boolean Operations Classes

Collaboration diagram for Boolean Operations Classes:



Classes

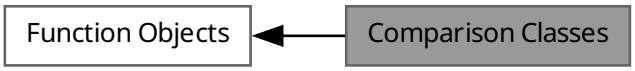
- struct std::logical_and< _Tp >
- struct std::logical_and< void >
- struct std::logical_not< _Tp >
- struct std::logical_not< void >
- struct std::logical_or< _Tp >
- struct std::logical_or< void >

3.15.4.7.1 Detailed Description

The library provides function objects for the logical operations: `&&`, ||, and `!`.

3.15.4.8 Comparison Classes

Collaboration diagram for Comparison Classes:



Classes

- struct std::equal_to< _Tp >
- struct std::greater< _Tp >
- struct std::greater< void >
- struct std::greater_equal< _Tp >
- struct std::greater_equal< void >
- struct std::less< _Tp >
- struct std::less_equal< _Tp >
- struct std::less_equal< void >
- struct std::not_equal_to< _Tp >
- struct std::not_equal_to< void >

3.15.4.8.1 Detailed Description

The library provides six wrapper functors for all the basic comparisons in C++, like <.

3.15.4.9 Hashes

Collaboration diagram for Hashes:

Classes

- struct std::hash< _Tp >
- struct std::hash< _Tp * >
- struct std::hash< bool >
- struct std::hash< char >
- struct std::hash< char16_t >
- struct std::hash< char32_t >
- struct std::hash< double >
- struct std::hash< float >
- struct std::hash< int >
- struct std::hash< long >
- struct std::hash< long double >
- struct std::hash< long long >
- struct std::hash< short >
- struct std::hash< signed char >
- struct std::hash< unsigned char >
- struct std::hash< unsigned int >
- struct std::hash< unsigned long >
- struct std::hash< unsigned long long >
- struct std::hash< unsigned short >
- struct std::hash< wchar_t >

Macros

- #define **_Cxx_hashtable_define_trivial_hash**(_Tp)

Variables

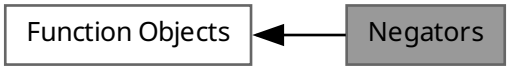
- template<typename _Tp, typename = void>
  constexpr bool **std::__is_hash_enabled_for**
- template<typename _Tp>
  constexpr bool **std::__is_hash_enabled_for**< **_Tp, __void_t**< **decltype(hash**< **_Tp** >()(**declval**< **_Tp** >()))> >

3.15.4.9.1 Detailed Description

Hashing functors taking a variable type and returning a `std::size_t`.

3.15.4.10 Negators

Collaboration diagram for Negators:



Classes

- class std::binary_negate< _Predicate >
- class std::unary_negate< _Predicate >

Functions

- template<typename _Predicate>
  constexpr unary_negate< _Predicate > std::not1 (const _Predicate &__pred)
- template<typename _Predicate>
  constexpr binary_negate< _Predicate > std::not2 (const _Predicate &__pred)

3.15.4.10.1 Detailed Description

The function templates `not1` and `not2` are function object adaptors, which each take a predicate functor and wrap it in an instance of `unary_negate` or `binary_negate`, respectively. Those classes are functors whose `operator()` evaluates the wrapped predicate function and then returns the negation of the result.
For example, given a vector of integers and a trivial predicate,

```
struct IntGreaterThanThree
  : public std::unary_function<int, bool>
{
    bool operator() (int x) const { return x > 3; }
};

std::find_if (v.begin(), v.end(), not1(IntGreaterThanThree()));
```

The call to `find_if` will locate the first index (i) of `v` for which `!(v[i] > 3)` is true.
The not1/unary_negate combination works on predicates taking a single argument. The not2/binary_negate combination works on predicates taking two arguments.

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

3.15.4.10.2 Function Documentation

not1()

```
template<typename _Predicate>
unary_negate< _Predicate > std::not1 (
            const _Predicate & __pred)  [inline], [constexpr]
```

One of the negation functors.

References not1().
Referenced by not1().

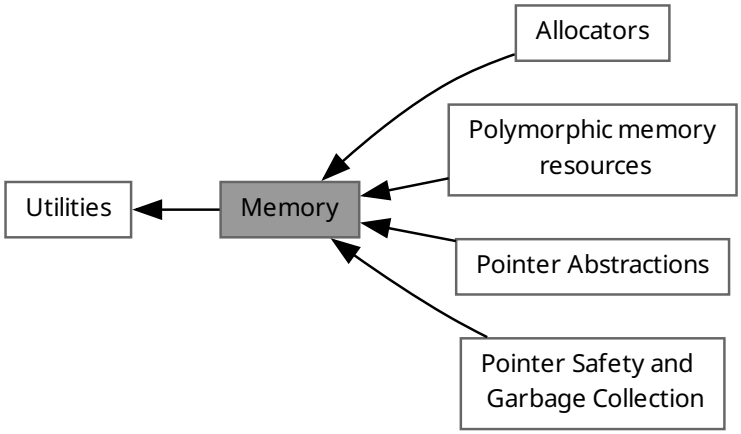
not2()

```
template<typename _Predicate>
binary_negate< _Predicate > std::not2 (
            const _Predicate & __pred)  [inline], [constexpr]
```

One of the negation functors.
References not2().
Referenced by not2().

3.15.5 Memory

Collaboration diagram for Memory:



Topics

- Allocators
- Pointer Abstractions
- Pointer Safety and Garbage Collection
- Polymorphic memory resources

Files

- file memory

Functions

- void * std::align (size_t __align, size_t __size, void *&__ptr, size_t &__space) noexcept
- template<size_t _Align, class _Tp>
  constexpr _Tp * std::assume_aligned (_Tp *__ptr) noexcept

- template<typename _InputIterator, typename _ForwardIterator>
  _ForwardIterator std::uninitialized_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __result)
- template<typename _InputIterator, typename _Size, typename _ForwardIterator>
  _ForwardIterator std::uninitialized_copy_n (_InputIterator __first, _Size __n, _ForwardIterator __result)
- template<typename _ForwardIterator>
  void std::uninitialized_default_construct (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _Size>
  _ForwardIterator std::uninitialized_default_construct_n (_ForwardIterator __first, _Size __count)
- template<typename _ForwardIterator, typename _Tp>
  void std::uninitialized_fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__x)
- template<typename _ForwardIterator, typename _Size, typename _Tp>
  _ForwardIterator std::uninitialized_fill_n (_ForwardIterator __first, _Size __n, const _Tp &__x)
- template<typename _InputIterator, typename _ForwardIterator>
  _ForwardIterator std::uninitialized_move (_InputIterator __first, _InputIterator __last, _ForwardIterator __result)
- template<typename _InputIterator, typename _Size, typename _ForwardIterator>
  pair< _InputIterator, _ForwardIterator > std::uninitialized_move_n (_InputIterator __first, _Size __count, _↩
  ForwardIterator __result)
- template<typename _ForwardIterator>
  void std::uninitialized_value_construct (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _Size>
  _ForwardIterator std::uninitialized_value_construct_n (_ForwardIterator __first, _Size __count)

3.15.5.1 Detailed Description

Components for memory allocation, deallocation, and management.

3.15.5.2 Function Documentation

align()

```
void * std::align (
            size_t __align,
            size_t __size,
            void *& __ptr,
            size_t & __space)  [inline], [noexcept]
```

Fit aligned storage in buffer.
This function tries to fit *__size* bytes of storage with alignment *__align* into the buffer *__ptr* of size *__space* bytes. If such a buffer fits then *__ptr* is changed to point to the first byte of the aligned storage and *__space* is reduced by the bytes used for alignment.
C++11 20.6.5 [ptr.align]

Parameters

__align	A fundamental or extended alignment value.
__size	Size of the aligned storage required.
__ptr	Pointer to a buffer of *__space* bytes.
__space	Size of the buffer pointed to by *__ptr*.

Returns

the updated pointer if the aligned storage fits, otherwise nullptr.

assume_aligned()

```
template<size_t _Align, class _Tp>
_Tp * std::assume_aligned (
            _Tp * __ptr)  [nodiscard], [constexpr], [noexcept]
```

Inform the compiler that a pointer is aligned.

Template Parameters

_Align	An alignment value (i.e. a power of two)
_Tp	An object type

Parameters

__ptr	A pointer that is aligned to _Align

C++20 20.10.6 [ptr.align]

uninitialized_copy()

```
template<typename _InputIterator, typename _ForwardIterator>
_ForwardIterator std::uninitialized_copy (
            _InputIterator __first,
            _InputIterator __last,
            _ForwardIterator __result)  [inline]
```

Copies the range [first,last) into result.

Parameters

__first	An input iterator.
__last	An input iterator.
__result	A forward iterator.

Returns

__result + (__last - __first)

Like std::copy, but does not require an initialized output range.
References to_address().
Referenced by __gnu_parallel::parallel_sort_mwms_pu(), and uninitialized_move().

uninitialized_copy_n()

```
template<typename _InputIterator, typename _Size, typename _ForwardIterator>
_ForwardIterator std::uninitialized_copy_n (
            _InputIterator __first,
            _Size __n,
            _ForwardIterator __result)  [inline]
```

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

Parameters

__first	An input iterator.

__n	The number of elements to copy.
__result	An output iterator.

Returns

__result + __n

Since

C++11

Like copy_n(), but does not require an initialized output range.
References __iterator_category().

uninitialized_default_construct()

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

Default-initializes objects in the range [first,last).

Parameters

__first	A forward iterator.
__last	A forward iterator.

Since

C++17

uninitialized_default_construct_n()

```
template<typename _ForwardIterator, typename _Size>
_ForwardIterator std::uninitialized_default_construct_n (
            _ForwardIterator __first,
            _Size __count)  [inline]
```

Default-initializes objects in the range [first,first+count).

Parameters

__first	A forward iterator.
__count	The number of objects to construct.

Returns

__first + __count

Since

C++17

uninitialized_fill()

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

Copies the value x into the range [first,last).

Parameters

__first	A forward iterator.
__last	A forward iterator.
__x	The source value.

Returns

Nothing.

Like std::fill, but does not require an initialized output range.
References to_address().

uninitialized_fill_n()

```
template<typename _ForwardIterator, typename _Size, typename _Tp>
_ForwardIterator std::uninitialized_fill_n (
            _ForwardIterator __first,
            _Size __n,
            const _Tp & __x)  [inline]
```

Copies the value x into the range [first,first+n).

Parameters

__first	A forward iterator.
__n	The number of copies to make.
__x	The source value.

Returns

__first + __n.

Like std::fill_n, but does not require an initialized output range.
References to_address().

uninitialized_move()

```
template<typename _InputIterator, typename _ForwardIterator>
_ForwardIterator std::uninitialized_move (
            _InputIterator __first,
            _InputIterator __last,
            _ForwardIterator __result)  [inline]
```

Move-construct from the range [first,last) into result.

Parameters

__first	An input iterator.
__last	An input iterator.
__result	An output iterator.

Returns

__result + (__first - __last)

Since

C++17

References uninitialized_copy().

uninitialized_move_n()

```
template<typename _InputIterator, typename _Size, typename _ForwardIterator>
pair< _InputIterator, _ForwardIterator > std::uninitialized_move_n (
            _InputIterator __first,
            _Size __count,
            _ForwardIterator __result)  [inline]
```

Move-construct from the range [first,first+count) into result.

Parameters

__first	An input iterator.
__count	The number of objects to initialize.
__result	An output iterator.

Returns

__result + __count

Since

C++17

uninitialized_value_construct()

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

Value-initializes objects in the range [first,last).

Parameters

__first	A forward iterator.
__last	A forward iterator.

Since

C++17

uninitialized_value_construct_n()

```
template<typename _ForwardIterator, typename _Size>
_ForwardIterator std::uninitialized_value_construct_n (
            _ForwardIterator __first,
            _Size __count)  [inline]
```

Value-initializes objects in the range [first,first+count).

Parameters

__first	A forward iterator.
__count	The number of objects to construct.

Returns

__result + __count

Since

C++17

3.15.5.3 Allocators

Collaboration diagram for Allocators:

Classes

- struct __gnu_cxx::__alloc_traits< _Alloc, typename >
- class __gnu_cxx::__mt_alloc< _Tp, _Poolp >
- class std::__new_allocator< _Tp >
- class __gnu_cxx::__pool_alloc< _Tp >
- class __gnu_cxx::_ExtPtr_allocator< _Tp >
- class std::allocator< _Tp >
- struct std::allocator_traits< _Alloc >
- struct std::allocator_traits< allocator< _Tp > >
- struct std::allocator_traits< allocator< void > >
- class __gnu_cxx::bitmap_allocator< _Tp >
- class __gnu_cxx::debug_allocator< _Alloc >
- class __gnu_cxx::malloc_allocator< _Tp >
- class __gnu_cxx::new_allocator< _Tp >
- class std::scoped_allocator_adaptor< _OuterAlloc, _InnerAllocs >
- class __gnu_cxx::throw_allocator_base< _Tp, _Cond >
- struct std::uses_allocator< typename, typename >

Typedefs

- template<typename _Tp>
  using std::__allocator_base

Functions

- template<typename _Tp, typename _Alloc, typename... _Args>
  constexpr _Tp **std::make_obj_using_allocator** (const _Alloc &__a, _Args &&... __args)
- template<typename _T1, typename _T2>
  constexpr bool operator== (const allocator< _T1 > &, const allocator< _T2 > &) noexcept
- template<typename _OutA1, typename _OutA2, typename... _InA>
  bool **operator==** (const scoped_allocator_adaptor< _OutA1, _InA... > &__a, const scoped_allocator_adaptor< _OutA2, _InA... > &__b) noexcept
- template<typename _Tp, typename _Alloc, typename... _Args>
  constexpr _Tp * **std::uninitialized_construct_using_allocator** (_Tp *__p, const _Alloc &__a, _Args &&... _↩
  _args)
- template<_Std_pair _Tp, typename _Alloc>
  constexpr auto **std::uses_allocator_construction_args** (const _Alloc &) noexcept
- template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>
  constexpr auto **std::uses_allocator_construction_args** (const _Alloc &, _Up &&, _Vp &&) noexcept
- template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>
  constexpr auto **std::uses_allocator_construction_args** (const _Alloc &, const pair< _Up, _Vp > &) noexcept
- template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>
  constexpr auto **std::uses_allocator_construction_args** (const _Alloc &, pair< _Up, _Vp > &&) noexcept
- template<typename _Tp, typename _Alloc, typename... _Args>
  requires (! _Std_pair<_Tp>)
  constexpr auto **std::uses_allocator_construction_args** (const _Alloc &__a, _Args &&... __args) noexcept
- template<_Std_pair _Tp, typename _Alloc, typename _Tuple1, typename _Tuple2>
  constexpr auto **std::uses_allocator_construction_args** (const _Alloc &__a, piecewise_construct_t, _Tuple1 &&__x, _Tuple2 &&__y) noexcept

3.15.5.3.1 Detailed Description

Classes encapsulating memory operations.

3.15.5.3.2 Typedef Documentation

__allocator_base

```
template<typename _Tp>
using std::__allocator_base
```

An alias to the base class for std::allocator.
Used to set the std::allocator base class to std::__new_allocator.

Template Parameters

_Tp	Type of allocated object.

3.15.5.3.3 Function Documentation

operator==()

```
template<typename _T1, typename _T2>
bool operator== (
```

```
const allocator< _T1 > & ,
const allocator< _T2 > & )  [related]
```

Equality comparison for std::allocator objects

Returns

true, for all std::allocator objects.

3.15.5.4 Pointer Abstractions

Collaboration diagram for Pointer Abstractions:

Classes

- class std::auto_ptr< _Tp >
- struct std::default_delete< _Tp >
- struct std::default_delete< _Tp[ ]>
- class std::enable_shared_from_this< _Tp >
- struct std::hash< shared_ptr< _Tp > >
- struct std::hash< unique_ptr< _Tp, _Dp > >
- struct std::owner_less< _Tp >
- struct std::owner_less< shared_ptr< _Tp > >
- struct std::owner_less< void >
- struct std::owner_less< weak_ptr< _Tp > >
- struct std::pointer_traits< _Ptr >
- struct std::pointer_traits< _Tp * >
- class std::shared_ptr< _Tp >
- class std::unique_ptr< _Tp, _Dp >
- class std::unique_ptr< _Tp[ ], _Dp >
- class std::weak_ptr< _Tp >

Functions

- template<typename _Del, typename _Tp, _Lock_policy _Lp>
  _Del * **std::get_deleter** (const __shared_ptr< _Tp, _Lp > &__p) noexcept
- template<typename _Del, typename _Tp>
  _Del * get_deleter (const shared_ptr< _Tp > &__p) noexcept
- template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
  constexpr bool std::operator< (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp, typename _Dp>
  constexpr bool std::operator< (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<typename _Tp, typename _Dp>
  constexpr bool std::operator< (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)

- template<typename _Ch, typename _Tr, typename _Tp, _Lock_policy _Lp>
  std::basic_ostream< _Ch, _Tr > & operator<< (std::basic_ostream< _Ch, _Tr > &__os, const __shared_ptr< _Tp, _Lp > &__p)
- template<typename _CharT, typename _Traits, typename _Tp, typename _Dp>
  basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const unique_ptr< _Tp, _Dp > &__p)
- template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
  constexpr bool std::operator<= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp, typename _Dp>
  constexpr bool std::operator<= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<typename _Tp, typename _Dp>
  constexpr bool std::operator<= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)
- template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
  requires three_way_comparable_with<typename unique_ptr<_Tp, _Dp>::pointer, typename unique_ptr<_Up, _Ep>::pointer>
  constexpr compare_three_way_result_t< typename unique_ptr< _Tp, _Dp >::pointer, typename unique_ptr< _Up, _Ep >::pointer > **std::operator<=>** (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp, typename _Dp>
  requires three_way_comparable<typename unique_ptr<_Tp, _Dp>::pointer>
  constexpr compare_three_way_result_t< typename unique_ptr< _Tp, _Dp >::pointer > **std::operator<=>** (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
  constexpr bool std::operator== (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp, typename _Dp>
  constexpr bool std::operator== (const unique_ptr< _Tp, _Dp > &__x, nullptr_t) noexcept
- template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
  constexpr bool std::operator> (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp, typename _Dp>
  constexpr bool std::operator> (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<typename _Tp, typename _Dp>
  constexpr bool std::operator> (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)
- template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
  constexpr bool std::operator>= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp, typename _Dp>
  constexpr bool std::operator>= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<typename _Tp, typename _Dp>
  bool std::operator>= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)
- template<typename _Tp, typename _Del>
  std::shared_ptr (unique_ptr< _Tp, _Del >) -> shared_ptr< _Tp >
- template<typename _Tp>
  std::shared_ptr (weak_ptr< _Tp >) -> shared_ptr< _Tp >
- template<typename _Tp, typename _Dp>
  enable_if<!__is_swappable< _Dp >::value >::type **std::swap** (unique_ptr< _Tp, _Dp > &, unique_ptr< _Tp, _Dp > &)=delete
- template<typename _Tp, typename _Dp>
  constexpr enable_if< __is_swappable< _Dp >::value >::type swap (unique_ptr< _Tp, _Dp > &__x, unique_ptr< _Tp, _Dp > &__y) noexcept
- template<typename _Tp>
  void swap (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b) noexcept
- template<typename _Tp>
  constexpr _Tp * std::to_address (_Tp *__ptr) noexcept
- template<typename _Ptr>
  constexpr auto std::to_address (const _Ptr &__ptr) noexcept

- template<typename _Tp>
  std::weak_ptr (shared_ptr< _Tp >) -> weak_ptr< _Tp >

- template<typename _Tp, _Lock_policy _Lp>
  bool std::atomic_is_lock_free (const __shared_ptr< _Tp, _Lp > *)
- template<typename _Tp>
  bool std::atomic_is_lock_free (const shared_ptr< _Tp > *__p)

- template<typename _Tp>
  shared_ptr< _Tp > std::atomic_load_explicit (const shared_ptr< _Tp > *__p, memory_order)
- template<typename _Tp>
  shared_ptr< _Tp > std::atomic_load (const shared_ptr< _Tp > *__p)
- template<typename _Tp, _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > std::atomic_load_explicit (const __shared_ptr< _Tp, _Lp > *__p, memory_order)
- template<typename _Tp, _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > std::atomic_load (const __shared_ptr< _Tp, _Lp > *__p)

- template<typename _Tp>
  void std::atomic_store_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r, memory_order)
- template<typename _Tp>
  void std::atomic_store (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r)
- template<typename _Tp, _Lock_policy _Lp>
  void std::atomic_store_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r, memory_order)
- template<typename _Tp, _Lock_policy _Lp>
  void std::atomic_store (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r)

- template<typename _Tp>
  shared_ptr< _Tp > std::atomic_exchange_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r, memory_order)
- template<typename _Tp>
  shared_ptr< _Tp > std::atomic_exchange (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r)
- template<typename _Tp, _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > std::atomic_exchange_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r, memory_order)
- template<typename _Tp, _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > std::atomic_exchange (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r)

- template<typename _Tp>
  bool std::atomic_compare_exchange_strong_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w, memory_order, memory_order)
- template<typename _Tp>
  bool std::atomic_compare_exchange_strong (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w)
- template<typename _Tp>
  bool std::atomic_compare_exchange_weak_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w, memory_order __success, memory_order __failure)
- template<typename _Tp>
  bool std::atomic_compare_exchange_weak (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w)
- template<typename _Tp, _Lock_policy _Lp>
  bool std::atomic_compare_exchange_strong_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w, memory_order, memory_order)

- template<typename _Tp, _Lock_policy _Lp>
  bool std::atomic_compare_exchange_strong (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *↩
  __v, __shared_ptr< _Tp, _Lp > __w)
- template<typename _Tp, _Lock_policy _Lp>
  bool std::atomic_compare_exchange_weak_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w, memory_order __success, memory_order __failure)
- template<typename _Tp, _Lock_policy _Lp>
  bool std::atomic_compare_exchange_weak (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w)

- template<typename _Tp, typename _Up>
  bool operator== (const shared_ptr< _Tp > &__a, const shared_ptr< _Up > &__b) noexcept
- template<typename _Tp>
  bool operator== (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp>
  void swap (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b) noexcept
- template<typename _Tp, typename _Up>
  shared_ptr< _Tp > static_pointer_cast (const shared_ptr< _Up > &__r) noexcept
- template<typename _Tp, typename _Up>
  shared_ptr< _Tp > const_pointer_cast (const shared_ptr< _Up > &__r) noexcept
- template<typename _Tp, typename _Up>
  shared_ptr< _Tp > dynamic_pointer_cast (const shared_ptr< _Up > &__r) noexcept
- template<typename _Tp, typename _Up>
  shared_ptr< _Tp > reinterpret_pointer_cast (const shared_ptr< _Up > &__r) noexcept
- template<typename _Tp, typename _Up>
  shared_ptr< _Tp > static_pointer_cast (shared_ptr< _Up > &&__r) noexcept
- template<typename _Tp, typename _Up>
  shared_ptr< _Tp > const_pointer_cast (shared_ptr< _Up > &&__r) noexcept
- template<typename _Tp, typename _Up>
  shared_ptr< _Tp > dynamic_pointer_cast (shared_ptr< _Up > &&__r) noexcept
- template<typename _Tp, typename _Up>
  shared_ptr< _Tp > reinterpret_pointer_cast (shared_ptr< _Up > &&__r) noexcept

- template<typename _Tp, typename _Alloc, typename... _Args>
  shared_ptr< _NonArray< _Tp > > allocate_shared (const _Alloc &__a, _Args &&... __args)
- template<typename _Tp, typename... _Args>
  shared_ptr< _NonArray< _Tp > > make_shared (_Args &&... __args)

3.15.5.4.1 Detailed Description

Smart pointers, etc.

3.15.5.4.2 Function Documentation

allocate_shared()

```
template<typename _Tp, typename _Alloc, typename...  _Args>
shared_ptr< _NonArray< _Tp > > allocate_shared (
            const _Alloc & __a,
            _Args &&...  __args)  [related]
```

Create an object that is owned by a shared_ptr.

Parameters

__a	An allocator.

__args	Arguments for the *_Tp* object's constructor.

Returns

A shared_ptr that owns the newly created object.

Exceptions

An	exception thrown from *_Alloc::allocate* or from the constructor of *_Tp*.

A copy of *__a* will be used to allocate memory for the shared_ptr and the new object.

atomic_compare_exchange_strong() [1/2]

```
template<typename _Tp, _Lock_policy _Lp>
bool std::atomic_compare_exchange_strong (
             __shared_ptr< _Tp, _Lp > * __p,
             __shared_ptr< _Tp, _Lp > * __v,
             __shared_ptr< _Tp, _Lp > __w)  [inline]
```

Atomic compare-and-swap for shared_ptr objects.

Parameters

_↩ _p	A non-null pointer to a shared_ptr object.
_↩ _v	A non-null pointer to a shared_ptr object.
_↩ _w	A non-null pointer to a shared_ptr object.

Returns

True if `*__p` was equivalent to `*__v`, false otherwise.

The memory order for failure shall not be `memory_order_release` or `memory_order_acq_rel`.
References move().

atomic_compare_exchange_strong() [2/2]

```
template<typename _Tp>
bool std::atomic_compare_exchange_strong (
             shared_ptr< _Tp > * __p,
             shared_ptr< _Tp > * __v,
             shared_ptr< _Tp > __w)  [inline]
```

Atomic compare-and-swap for shared_ptr objects.

Parameters

_↩ _p	A non-null pointer to a shared_ptr object.
_↩ _v	A non-null pointer to a shared_ptr object.

_↩ _w	A non-null pointer to a shared_ptr object.

Returns

True if ∗__p was equivalent to ∗__v, false otherwise.

The memory order for failure shall not be memory_order_release or memory_order_acq_rel.
References move().

atomic_compare_exchange_strong_explicit() [1/2]

```
template<typename _Tp, _Lock_policy _Lp>
bool std::atomic_compare_exchange_strong_explicit (
             __shared_ptr< _Tp, _Lp > * __p,
             __shared_ptr< _Tp, _Lp > * __v,
             __shared_ptr< _Tp, _Lp > __w,
             memory_order ,
             memory_order )
```

Atomic compare-and-swap for shared_ptr objects.

Parameters

_↩ _p	A non-null pointer to a shared_ptr object.
_↩ _v	A non-null pointer to a shared_ptr object.
_↩ _w	A non-null pointer to a shared_ptr object.

Returns

True if ∗__p was equivalent to ∗__v, false otherwise.

The memory order for failure shall not be memory_order_release or memory_order_acq_rel.
References move().

atomic_compare_exchange_strong_explicit() [2/2]

```
template<typename _Tp>
bool std::atomic_compare_exchange_strong_explicit (
             shared_ptr< _Tp > * __p,
             shared_ptr< _Tp > * __v,
             shared_ptr< _Tp > __w,
             memory_order ,
             memory_order )
```

Atomic compare-and-swap for shared_ptr objects.

Parameters

_↩ _p	A non-null pointer to a shared_ptr object.
_↩ _v	A non-null pointer to a shared_ptr object.

_↩ _w	A non-null pointer to a shared_ptr object.

Returns

True if ∗__p was equivalent to ∗__v, false otherwise.

The memory order for failure shall not be memory_order_release or memory_order_acq_rel.
References move().

atomic_compare_exchange_weak() [1/2]

```
template<typename _Tp, _Lock_policy _Lp>
bool std::atomic_compare_exchange_weak (
             __shared_ptr< _Tp, _Lp > * __p,
             __shared_ptr< _Tp, _Lp > * __v,
             __shared_ptr< _Tp, _Lp > __w)  [inline]
```

Atomic compare-and-swap for shared_ptr objects.

Parameters

_↩ _p	A non-null pointer to a shared_ptr object.
_↩ _v	A non-null pointer to a shared_ptr object.
_↩ _w	A non-null pointer to a shared_ptr object.

Returns

True if ∗__p was equivalent to ∗__v, false otherwise.

The memory order for failure shall not be memory_order_release or memory_order_acq_rel.
References move().

atomic_compare_exchange_weak() [2/2]

```
template<typename _Tp>
bool std::atomic_compare_exchange_weak (
             shared_ptr< _Tp > * __p,
             shared_ptr< _Tp > * __v,
             shared_ptr< _Tp > __w)  [inline]
```

Atomic compare-and-swap for shared_ptr objects.

Parameters

_↩ _p	A non-null pointer to a shared_ptr object.
_↩ _v	A non-null pointer to a shared_ptr object.
_↩ _w	A non-null pointer to a shared_ptr object.

Returns

True if ∗__p was equivalent to ∗__v, false otherwise.

The memory order for failure shall not be memory_order_release or memory_order_acq_rel.
References move().

atomic_compare_exchange_weak_explicit() [1/2]

```
template<typename _Tp, _Lock_policy _Lp>
bool std::atomic_compare_exchange_weak_explicit (
             __shared_ptr< _Tp, _Lp > * __p,
             __shared_ptr< _Tp, _Lp > * __v,
             __shared_ptr< _Tp, _Lp > __w,
             memory_order __success,
             memory_order __failure)  [inline]
```

Atomic compare-and-swap for shared_ptr objects.

Parameters

_↩ _p	A non-null pointer to a shared_ptr object.
_↩ _v	A non-null pointer to a shared_ptr object.
_↩ _w	A non-null pointer to a shared_ptr object.

Returns

True if ∗__p was equivalent to ∗__v, false otherwise.

The memory order for failure shall not be memory_order_release or memory_order_acq_rel.
References move().

atomic_compare_exchange_weak_explicit() [2/2]

```
template<typename _Tp>
bool std::atomic_compare_exchange_weak_explicit (
             shared_ptr< _Tp > * __p,
             shared_ptr< _Tp > * __v,
             shared_ptr< _Tp > __w,
             memory_order __success,
             memory_order __failure)  [inline]
```

Atomic compare-and-swap for shared_ptr objects.

Parameters

_↩ _p	A non-null pointer to a shared_ptr object.
_↩ _v	A non-null pointer to a shared_ptr object.
_↩ _w	A non-null pointer to a shared_ptr object.

Returns

True if *__p was equivalent to *__v, false otherwise.

The memory order for failure shall not be memory_order_release or memory_order_acq_rel.
References move().

atomic_exchange() [1/2]

```
template<typename _Tp, _Lock_policy _Lp>
__shared_ptr< _Tp, _Lp > std::atomic_exchange (
             __shared_ptr< _Tp, _Lp > * __p,
             __shared_ptr< _Tp, _Lp > __r)  [inline]
```

Atomic exchange for shared_ptr objects.

Parameters

_↩ _p	A non-null pointer to a shared_ptr object.
_↩ _r	New value to store in *__p.

Returns

The original value of *__p

References move().

atomic_exchange() [2/2]

```
template<typename _Tp>
shared_ptr< _Tp > std::atomic_exchange (
             shared_ptr< _Tp > * __p,
             shared_ptr< _Tp > __r)  [inline]
```

Atomic exchange for shared_ptr objects.

Parameters

_↩ _p	A non-null pointer to a shared_ptr object.
_↩ _r	New value to store in *__p.

Returns

The original value of *__p

References move().

atomic_exchange_explicit() [1/2]

```
template<typename _Tp, _Lock_policy _Lp>
__shared_ptr< _Tp, _Lp > std::atomic_exchange_explicit (
             __shared_ptr< _Tp, _Lp > * __p,
             __shared_ptr< _Tp, _Lp > __r,
             memory_order )  [inline]
```

Atomic exchange for shared_ptr objects.

Parameters

_↩ _p	A non-null pointer to a shared_ptr object.
_↩ _r	New value to store in *__p.

Returns

The original value of *__p

atomic_exchange_explicit() [2/2]

```
template<typename _Tp>
shared_ptr< _Tp > std::atomic_exchange_explicit (
            shared_ptr< _Tp > * __p,
            shared_ptr< _Tp > __r,
            memory_order )  [inline]
```

Atomic exchange for shared_ptr objects.

Parameters

_↩ _p	A non-null pointer to a shared_ptr object.
_↩ _r	New value to store in *__p.

Returns

The original value of *__p

atomic_is_lock_free() [1/2]

```
template<typename _Tp, _Lock_policy _Lp>
bool std::atomic_is_lock_free (
            const __shared_ptr< _Tp, _Lp > * )  [inline]
```

Report whether shared_ptr atomic operations are lock-free.

Parameters

_↩ _p	A non-null pointer to a shared_ptr object.

Returns

True if atomic access to *__p is lock-free, false otherwise.

atomic_is_lock_free() [2/2]

```
template<typename _Tp>
bool std::atomic_is_lock_free (
            const shared_ptr< _Tp > * __p)  [inline]
```

Report whether shared_ptr atomic operations are lock-free.

Parameters

_↩ _p	A non-null pointer to a shared_ptr object.

Returns

True if atomic access to *__p is lock-free, false otherwise.

atomic_load() [1/2]

```
template<typename _Tp, _Lock_policy _Lp>
__shared_ptr< _Tp, _Lp > std::atomic_load (
            const __shared_ptr< _Tp, _Lp > * __p)  [inline]
```

Atomic load for shared_ptr objects.

Parameters

_↩ _p	A non-null pointer to a shared_ptr object.

Returns

*__p

The memory order shall not be memory_order_release or memory_order_acq_rel.

atomic_load() [2/2]

```
template<typename _Tp>
shared_ptr< _Tp > std::atomic_load (
            const shared_ptr< _Tp > * __p)  [inline]
```

Atomic load for shared_ptr objects.

Parameters

_↩ _p	A non-null pointer to a shared_ptr object.

Returns

*__p

The memory order shall not be memory_order_release or memory_order_acq_rel.

atomic_load_explicit() [1/2]

```
template<typename _Tp, _Lock_policy _Lp>
__shared_ptr< _Tp, _Lp > std::atomic_load_explicit (
            const __shared_ptr< _Tp, _Lp > * __p,
            memory_order )  [inline]
```

Atomic load for shared_ptr objects.

Parameters

_↩ _p	A non-null pointer to a shared_ptr object.

Returns

*__p

The memory order shall not be memory_order_release or memory_order_acq_rel.

atomic_load_explicit() [2/2]

```
template<typename _Tp>
shared_ptr< _Tp > std::atomic_load_explicit (
            const shared_ptr< _Tp > * __p,
            memory_order )  [inline]
```

Atomic load for shared_ptr objects.

Parameters

_↩ _p	A non-null pointer to a shared_ptr object.

Returns

*__p

The memory order shall not be memory_order_release or memory_order_acq_rel.

atomic_store() [1/2]

```
template<typename _Tp, _Lock_policy _Lp>
void std::atomic_store (
            __shared_ptr< _Tp, _Lp > * __p,
            __shared_ptr< _Tp, _Lp > __r)  [inline]
```

Atomic store for shared_ptr objects.

Parameters

_↩ _p	A non-null pointer to a shared_ptr object.
_↩ _r	The value to store.

The memory order shall not be memory_order_acquire or memory_order_acq_rel.
References move().

atomic_store() [2/2]

```
template<typename _Tp>
void std::atomic_store (
            shared_ptr< _Tp > * __p,
            shared_ptr< _Tp > __r)  [inline]
```

Atomic store for shared_ptr objects.

Parameters

_↩ _p	A non-null pointer to a shared_ptr object.
_↩ _r	The value to store.

The memory order shall not be `memory_order_acquire` or `memory_order_acq_rel`.
References move().

atomic_store_explicit() [1/2]

```
template<typename _Tp, _Lock_policy _Lp>
void std::atomic_store_explicit (
             __shared_ptr< _Tp, _Lp > * __p,
             __shared_ptr< _Tp, _Lp > __r,
             memory_order )  [inline]
```

Atomic store for shared_ptr objects.

Parameters

_↩ _p	A non-null pointer to a shared_ptr object.
_↩ _r	The value to store.

The memory order shall not be `memory_order_acquire` or `memory_order_acq_rel`.

atomic_store_explicit() [2/2]

```
template<typename _Tp>
void std::atomic_store_explicit (
             shared_ptr< _Tp > * __p,
             shared_ptr< _Tp > __r,
             memory_order )  [inline]
```

Atomic store for shared_ptr objects.

Parameters

_↩ _p	A non-null pointer to a shared_ptr object.
_↩ _r	The value to store.

The memory order shall not be `memory_order_acquire` or `memory_order_acq_rel`.

const_pointer_cast() [1/2]

```
template<typename _Tp, typename _Up>
shared_ptr< _Tp > const_pointer_cast (
             const shared_ptr< _Up > & __r)  [related]
```

Convert type of `shared_ptr`, via `const_cast`

const_pointer_cast() [2/2]

```
template<typename _Tp, typename _Up>
shared_ptr< _Tp > const_pointer_cast (
            shared_ptr< _Up > && __r)  [related]
```

Convert type of `shared_ptr` rvalue, via `const_cast`

Since

C++20

dynamic_pointer_cast() [1/2]

```
template<typename _Tp, typename _Up>
shared_ptr< _Tp > dynamic_pointer_cast (
            const shared_ptr< _Up > & __r)  [related]
```

Convert type of `shared_ptr`, via `dynamic_cast`

dynamic_pointer_cast() [2/2]

```
template<typename _Tp, typename _Up>
shared_ptr< _Tp > dynamic_pointer_cast (
            shared_ptr< _Up > && __r)  [related]
```

Convert type of `shared_ptr` rvalue, via `dynamic_cast`

Since

C++20

get_deleter()

```
template<typename _Del, typename _Tp>
_Del * get_deleter (
            const shared_ptr< _Tp > & __p)  [related]
```

20.7.2.2.10 shared_ptr get_deleter
If `__p` has a deleter of type `_Del`, return a pointer to it.

make_shared()

```
template<typename _Tp, typename...  _Args>
shared_ptr< _NonArray< _Tp > > make_shared (
            _Args &&...  __args)  [related]
```

Create an object that is owned by a shared_ptr.

Parameters

__args	Arguments for the *_Tp* object's constructor.

Returns

A shared_ptr that owns the newly created object.

Exceptions

std::bad_alloc,or	an exception thrown from the constructor of *_Tp*.

operator<() [1/3]

```
template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
bool std::operator< (
            const unique_ptr< _Tp, _Dp > & __x,
            const unique_ptr< _Up, _Ep > & __y)  [inline], [nodiscard], [constexpr]
```

Relational operator for unique_ptr objects, compares the owned pointers.
References std::unique_ptr< _Tp, _Dp >::get().

operator<() [2/3]

```
template<typename _Tp, typename _Dp>
bool std::operator< (
            const unique_ptr< _Tp, _Dp > & __x,
            nullptr_t )  [inline], [nodiscard], [constexpr]
```

unique_ptr comparison with nullptr
References std::unique_ptr< _Tp, _Dp >::get().

operator<() [3/3]

```
template<typename _Tp, typename _Dp>
bool std::operator< (
            nullptr_t ,
            const unique_ptr< _Tp, _Dp > & __x)  [inline], [nodiscard], [constexpr]
```

unique_ptr comparison with nullptr
References std::unique_ptr< _Tp, _Dp >::get().

operator<<() [1/2]

```
template<typename _Ch, typename _Tr, typename _Tp, _Lock_policy _Lp>
std::basic_ostream< _Ch, _Tr > & operator<< (
            std::basic_ostream< _Ch, _Tr > & __os,
            const __shared_ptr< _Tp, _Lp > & __p)  [related]
```

Write the stored pointer to an ostream.

operator<<() [2/2]

```
template<typename _CharT, typename _Traits, typename _Tp, typename _Dp>
basic_ostream< _CharT, _Traits > & operator<< (
            basic_ostream< _CharT, _Traits > & __os,
            const unique_ptr< _Tp, _Dp > & __p)  [related]
```

Stream output operator for unique_ptr.

Since

C++20

operator<=() [1/3]

```
template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
bool std::operator<= (
            const unique_ptr< _Tp, _Dp > & __x,
            const unique_ptr< _Up, _Ep > & __y)  [inline], [nodiscard], [constexpr]
```

Relational operator for unique_ptr objects, compares the owned pointers.

operator<=() [2/3]

```
template<typename _Tp, typename _Dp>
bool std::operator<= (
            const unique_ptr< _Tp, _Dp > & __x,
            nullptr_t )  [inline], [nodiscard], [constexpr]
```

unique_ptr comparison with nullptr

operator<=() [3/3]

```
template<typename _Tp, typename _Dp>
bool std::operator<= (
            nullptr_t ,
            const unique_ptr< _Tp, _Dp > & __x)  [inline], [nodiscard], [constexpr]
```

unique_ptr comparison with nullptr

operator==() [1/4]

```
template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
bool std::operator== (
            const unique_ptr< _Tp, _Dp > & __x,
            const unique_ptr< _Up, _Ep > & __y)  [inline], [nodiscard], [constexpr]
```

Equality operator for unique_ptr objects, compares the owned pointers.
References std::unique_ptr< _Tp, _Dp >::get().

operator==() [2/4]

```
template<typename _Tp, typename _Dp>
bool std::operator== (
            const unique_ptr< _Tp, _Dp > & __x,
            nullptr_t )  [inline], [nodiscard], [constexpr], [noexcept]
```

unique_ptr comparison with nullptr

operator==() [3/4]

```
template<typename _Tp, typename _Up>
bool operator== (
            const shared_ptr< _Tp > & __a,
            const shared_ptr< _Up > & __b)  [related]
```

Equality operator for shared_ptr objects, compares the stored pointers.

operator==() [4/4]

```
template<typename _Tp>
bool operator== (
            const shared_ptr< _Tp > & __a,
            nullptr_t )  [related]
```

shared_ptr comparison with nullptr

operator>() [1/3]

```
template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
bool std::operator> (
            const unique_ptr< _Tp, _Dp > & __x,
            const unique_ptr< _Up, _Ep > & __y)  [inline], [nodiscard], [constexpr]
```

Relational operator for unique_ptr objects, compares the owned pointers.

operator>() [2/3]

```
template<typename _Tp, typename _Dp>
bool std::operator> (
            const unique_ptr< _Tp, _Dp > & __x,
            nullptr_t ) [inline], [nodiscard], [constexpr]
```

unique_ptr comparison with nullptr
References std::unique_ptr< _Tp, _Dp >::get().

operator>() [3/3]

```
template<typename _Tp, typename _Dp>
bool std::operator> (
            nullptr_t ,
            const unique_ptr< _Tp, _Dp > & __x) [inline], [nodiscard], [constexpr]
```

unique_ptr comparison with nullptr
References std::unique_ptr< _Tp, _Dp >::get().

operator>=() [1/3]

```
template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
bool std::operator>= (
            const unique_ptr< _Tp, _Dp > & __x,
            const unique_ptr< _Up, _Ep > & __y) [inline], [nodiscard], [constexpr]
```

Relational operator for unique_ptr objects, compares the owned pointers.

operator>=() [2/3]

```
template<typename _Tp, typename _Dp>
bool std::operator>= (
            const unique_ptr< _Tp, _Dp > & __x,
            nullptr_t ) [inline], [nodiscard], [constexpr]
```

unique_ptr comparison with nullptr

operator>=() [3/3]

```
template<typename _Tp, typename _Dp>
bool std::operator>= (
            nullptr_t ,
            const unique_ptr< _Tp, _Dp > & __x) [inline], [nodiscard]
```

unique_ptr comparison with nullptr

reinterpret_pointer_cast() [1/2]

```
template<typename _Tp, typename _Up>
shared_ptr< _Tp > reinterpret_pointer_cast (
            const shared_ptr< _Up > & __r) [related]
```

Convert type of `shared_ptr`, via `reinterpret_cast`

Since

C++17

reinterpret_pointer_cast() [2/2]

```
template<typename _Tp, typename _Up>
shared_ptr< _Tp > reinterpret_pointer_cast (
            shared_ptr< _Up > && __r)  [related]
```

Convert type of `shared_ptr` rvalue, via `reinterpret_cast`

Since

C++20

static_pointer_cast() [1/2]

```
template<typename _Tp, typename _Up>
shared_ptr< _Tp > static_pointer_cast (
            const shared_ptr< _Up > & __r)  [related]
```

Convert type of `shared_ptr`, via `static_cast`

static_pointer_cast() [2/2]

```
template<typename _Tp, typename _Up>
shared_ptr< _Tp > static_pointer_cast (
            shared_ptr< _Up > && __r)  [related]
```

Convert type of `shared_ptr` rvalue, via `static_cast`

Since

C++20

swap() [1/3]

```
template<typename _Tp>
void swap (
            shared_ptr< _Tp > & __a,
            shared_ptr< _Tp > & __b)  [related]
```

Swap overload for shared_ptr.

swap() [2/3]

```
template<typename _Tp, typename _Dp>
enable_if< __is_swappable< _Dp >::value >::type swap (
            unique_ptr< _Tp, _Dp > & __x,
            unique_ptr< _Tp, _Dp > & __y)  [related]
```

Swap overload for unique_ptr.

swap() [3/3]

```
template<typename _Tp>
void swap (
            weak_ptr< _Tp > & __a,
            weak_ptr< _Tp > & __b)  [related]
```

Swap overload for weak_ptr.

to_address() [1/2]

```
template<typename _Tp>
_Tp * std::to_address (
            _Tp * __ptr)  [constexpr], [noexcept]
```

Obtain address referenced by a pointer to an object.

Parameters

__ptr	A pointer to an object

Returns

`__ptr`

Referenced by std::basic_string< _CharT >::assign(), find(), to_address(), uninitialized_copy(), uninitialized_fill(), and uninitialized_fill_n().

to_address() [2/2]

```
template<typename _Ptr>
auto std::to_address (
            const _Ptr & __ptr)  [constexpr], [noexcept]
```

Obtain address referenced by a pointer to an object.

Parameters

__ptr	A pointer to an object

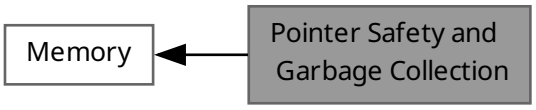
Returns

`pointer_traits<_Ptr>::to_address(__ptr)` if that expression is well-formed, otherwise `to_address(__ptr.operator->())`.

References to_address().

3.15.5.5 Pointer Safety and Garbage Collection

Collaboration diagram for Pointer Safety and Garbage Collection:



Enumerations

- enum class std::pointer_safety { **relaxed** , **preferred** , **strict** }

Functions

- void std::declare_no_pointers (char *, size_t)
- void std::declare_reachable (void *)
- pointer_safety std::get_pointer_safety () noexcept
- void std::undeclare_no_pointers (char *, size_t)
- template<typename _Tp>
  _Tp * std::undeclare_reachable (_Tp *__p)

3.15.5.5.1 Detailed Description

Utilities to assist with garbage collection in an implementation that supports *strict pointer safety*. This implementation only supports *relaxed pointer safety* and so these functions have no effect.
C++11 20.6.4 [util.dynamic.safety], Pointer safety

3.15.5.5.2 Enumeration Type Documentation

pointer_safety

```
enum class std::pointer_safety  [strong]
```

Constants representing the different types of pointer safety.

3.15.5.5.3 Function Documentation

declare_no_pointers()

```
void std::declare_no_pointers (
            char * ,
            size_t )  [inline]
```

Inform a garbage collector that a region of memory need not be traced.

declare_reachable()

```
void std::declare_reachable (
            void * )  [inline]
```

Inform a garbage collector that an object is still in use.

get_pointer_safety()

```
pointer_safety std::get_pointer_safety ()  [inline], [noexcept]
```

The type of pointer safety supported by the implementation.

undeclare_no_pointers()

```
void std::undeclare_no_pointers (
            char * ,
            size_t )  [inline]
```

Unregister a range previously registered with declare_no_pointers.

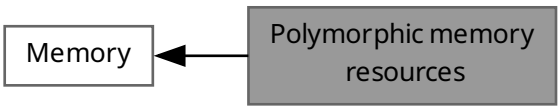
undeclare_reachable()

```
template<typename _Tp>
_Tp * std::undeclare_reachable (
            _Tp * __p)  [inline]
```

Unregister an object previously registered with declare_reachable.

3.15.5.6 Polymorphic memory resources

Collaboration diagram for Polymorphic memory resources:



Files

- file memory_resource

Classes

- struct std::allocator_traits< pmr::polymorphic_allocator< _Tp > >
- class std::pmr::memory_resource
- class std::pmr::monotonic_buffer_resource
- class std::pmr::polymorphic_allocator< _Tp >
- struct std::pmr::pool_options
- class std::pmr::unsynchronized_pool_resource

Functions

- memory_resource * std::pmr::new_delete_resource () noexcept

3.15.5.6.1 Detailed Description

Since

C++17

Memory resources are classes that implement the `std::pmr::memory_resource` interface for allocating and deallocating memory. Unlike traditional C++ allocators, memory resources are not value types and are used via pointers to the abstract base class. They are only responsible for allocating and deallocating, not for construction and destruction of objects. As a result, memory resources just allocate raw memory as type `void*` and are not templates that allocate/deallocate and construct/destroy a specific type.
The class template `std::pmr::polymorphic_allocator` is an allocator that uses a memory resource for its allocations.

3.15.5.6.2 Function Documentation

new_delete_resource()

```
memory_resource * std::pmr::new_delete_resource ()  [nodiscard], [noexcept]
```

A pmr::memory_resource that uses `new` to allocate memory.

Since

C++17

3.15.6 Metaprogramming

Collaboration diagram for Metaprogramming:

Classes

- struct std::tr2::__reflection_typelist< _Elements >
- struct std::tr2::__reflection_typelist< _First, _Rest... >
- struct std::tr2::__reflection_typelist<>
- struct std::add_const< _Tp >
- struct std::add_cv< _Tp >
- struct std::add_lvalue_reference< _Tp >
- struct std::add_pointer< _Tp >
- struct std::add_rvalue_reference< _Tp >
- struct std::add_volatile< _Tp >
- struct std::aligned_storage< _Len, _Align >
- struct std::aligned_union< _Len, _Types >
- struct std::alignment_of< _Tp >
- struct std::tr2::bases< _Tp >
- struct std::common_type< _Tp >
- struct std::conditional< _Cond, _Iftrue, _Iffalse >
- struct std::decay< _Tp >
- struct std::tr2::direct_bases< _Tp >
- struct std::enable_if< bool, _Tp >
- struct std::extent< typename, _Uint >
- struct std::has_virtual_destructor< _Tp >
- struct std::integral_constant< _Tp, __v >
- struct std::is_abstract< _Tp >
- struct std::is_arithmetic< _Tp >
- struct std::is_array< _Tp >
- struct std::is_assignable< _Tp, _Up >
- struct std::is_base_of< _Base, _Derived >
- struct std::is_class< _Tp >
- struct std::is_compound< _Tp >
- struct std::is_const< _Tp >
- struct std::is_constructible< _Tp, _Args >
- struct std::is_copy_assignable< _Tp >
- struct std::is_copy_constructible< _Tp >
- struct std::is_default_constructible< _Tp >
- struct std::is_destructible< _Tp >
- struct std::is_empty< _Tp >
- struct std::is_enum< _Tp >

- struct std::is_floating_point< _Tp >
- struct std::is_function< _Tp >
- struct std::is_fundamental< _Tp >
- struct std::is_integral< _Tp >
- struct std::is_layout_compatible< _Tp, _Up >
- struct std::is_literal_type< _Tp >
- struct std::is_lvalue_reference< typename >
- struct std::is_member_function_pointer< _Tp >
- struct std::is_member_object_pointer< _Tp >
- struct std::is_member_pointer< _Tp >
- struct std::is_move_assignable< _Tp >
- struct std::is_move_constructible< _Tp >
- struct std::is_nothrow_assignable< _Tp, _Up >
- struct std::is_nothrow_constructible< _Tp, _Args >
- struct std::is_nothrow_copy_assignable< _Tp >
- struct std::is_nothrow_copy_constructible< _Tp >
- struct std::is_nothrow_default_constructible< _Tp >
- struct std::is_nothrow_destructible< _Tp >
- struct std::is_nothrow_move_assignable< _Tp >
- struct std::is_nothrow_move_constructible< _Tp >
- struct std::is_object< _Tp >
- struct std::is_pod< _Tp >
- struct std::is_pointer< _Tp >
- struct std::is_pointer_interconvertible_base_of< _Base, _Derived >
- struct std::is_polymorphic< _Tp >
- struct std::is_reference< _Tp >
- struct std::is_rvalue_reference< typename >
- struct std::is_same< _Tp, _Up >
- struct std::is_scalar< _Tp >
- struct std::is_signed< _Tp >
- struct std::is_standard_layout< _Tp >
- struct std::is_trivial< _Tp >
- struct std::is_trivially_assignable< _Tp, _Up >
- struct std::is_trivially_constructible< _Tp, _Args >
- struct std::is_trivially_copy_assignable< _Tp >
- struct std::is_trivially_copy_constructible< _Tp >
- struct std::is_trivially_copyable< _Tp >
- struct std::is_trivially_default_constructible< _Tp >
- struct std::is_trivially_destructible< _Tp >
- struct std::is_trivially_move_assignable< _Tp >
- struct std::is_trivially_move_constructible< _Tp >
- struct std::is_union< _Tp >
- struct std::is_unsigned< _Tp >
- struct std::is_void< _Tp >
- struct std::is_volatile< _Tp >
- struct std::make_signed< _Tp >
- struct std::make_unsigned< _Tp >
- struct std::rank< _Tp >
- struct std::remove_all_extents< _Tp >
- struct std::remove_const< _Tp >
- struct std::remove_cv< _Tp >

- struct std::remove_extent< _Tp >
- struct std::remove_pointer< _Tp >
- struct std::remove_reference< _Tp >
- struct std::remove_volatile< _Tp >
- struct std::result_of< _Signature >
- struct std::underlying_type< _Tp >

Typedefs

- template<bool _Cond, typename _If, typename _Else>
  using **std::__conditional_t**
- template<bool _Cond, typename _Tp = void>
  using **std::__enable_if_t**
- template<typename _ToElementType, typename _FromElementType>
  using **std::__is_array_convertible**
- using **std::_Build_index_tuple< _Num >::__type**
- template<typename, size_t... _Indices>
  using **std::_Build_index_tuple< _Num >::_IdxTuple**
- template<typename _Tp>
  using std::add_lvalue_reference_t
- template<typename _Tp>
  using std::add_pointer_t
- template<typename _Tp>
  using std::add_rvalue_reference_t
- template<size_t _Len, size_t _Align = __aligned_storage_default_alignment(_Len)>
  using std::aligned_storage_t
- template<size_t _Len, typename... _Types>
  using **std::aligned_union_t**
- template<typename... _Tp>
  using std::common_type_t
- template<bool _Cond, typename _Iftrue, typename _Iffalse>
  using std::conditional_t
- template<typename _Tp>
  using std::decay_t
- template<bool _Cond, typename _Tp = void>
  using std::enable_if_t
- using std::false_type
- template<typename _Tp>
  using std::make_signed_t
- template<typename _Tp>
  using std::make_unsigned_t
- template<typename _Tp>
  using std::remove_all_extents_t
- template<typename _Tp>
  using std::remove_extent_t
- template<typename _Tp>
  using std::remove_pointer_t
- template<typename _Tp>
  using std::remove_reference_t
- template<typename _Tp>
  using std::result_of_t
- using std::true_type

- template<typename _Tp, typename>
  using **std::__conditional< bool >::type**
- template<typename, typename _Up>
  using **std::__conditional< false >::type**
- using **std::__is_implicitly_default_constructible_impl< _Tp >::type**
- using **std::add_const< _Tp >::type**
- using **std::add_cv< _Tp >::type**
- using **std::add_lvalue_reference< _Tp >::type**
- using **std::add_pointer< _Tp >::type**
- using **std::add_rvalue_reference< _Tp >::type**
- using **std::add_volatile< _Tp >::type**
- using std::aligned_union< _Len, _Types >::type
- using **std::common_reference< _Tp0 >::type**
- using **std::conditional< _Cond, _Iftrue, _Iffalse >::type**
- using **std::conditional< false, _Iftrue, _Iffalse >::type**
- using **std::decay< _Tp >::type**
- using **std::decay< _Tp & >::type**
- using **std::decay< _Tp && >::type**
- using **std::enable_if< true, _Tp >::type**
- using **std::integral_constant< _Tp, __v >::type**
- using **std::make_signed< _Tp >::type**
- using **std::make_unsigned< _Tp >::type**
- using **std::remove_all_extents< _Tp >::type**
- using **std::remove_const< _Tp >::type**
- using **std::remove_const< _Tp const >::type**
- using **std::remove_cv< _Tp >::type**
- using **std::remove_extent< _Tp >::type**
- using **std::remove_pointer< _Tp >::type**
- using **std::remove_reference< _Tp >::type**
- using **std::remove_volatile< _Tp >::type**
- using **std::remove_volatile< _Tp volatile >::type**
- template<typename _Tp>
  using std::underlying_type_t
- using **std::integral_constant< _Tp, __v >::value_type**

Functions

- template<typename _Tp>
  static void **std::__do_is_implicitly_default_constructible_impl::__helper** (const _Tp &)
- static false_type **std::__do_is_implicitly_default_constructible_impl::__test** (...)
- template<typename _Tp>
  static true_type **std::__do_is_implicitly_default_constructible_impl::__test** (const _Tp &, decltype(__helper< const _Tp & >({})) *=0)
- constexpr **std::integral_constant< _Tp, __v >::operator value_type** () const noexcept
- template<typename _Tp>
  constexpr _Require< __not_< __is_tuple_like< _Tp > >, is_move_constructible< _Tp >, is_move_assignable< _Tp > > std::swap (_Tp &, _Tp &) noexcept(__and_< is_nothrow_move_constructible< _Tp >, is_nothrow_move_assignable< _Tp > >::value)
- template<typename _Tp, size_t _Nm>
  constexpr __enable_if_t< __is_swappable< _Tp >::value > std::swap (_Tp(&__a)[_Nm], _Tp(&__b)[_Nm]) noexcept(__is_nothrow_swappable< _Tp >::value)

Variables

- unsigned char **std::aligned_storage< _Len, _Align >::type::__data** [_Len]
- template<typename _Tp, typename... _Args>
  constexpr bool **std::__is_nothrow_new_constructible**
- static const size_t **std::__strictest_alignment< _Types >::_S_alignment**
- static const size_t **std::__strictest_alignment< _Tp, _Types... >::_S_alignment**
- static const size_t **std::__strictest_alignment< _Types >::_S_size**
- static const size_t **std::__strictest_alignment< _Tp, _Types... >::_S_size**
- template<size_t _Len, typename... _Types>
  const size_t **std::aligned_union< _Len, _Types... >::alignment_value**
- static const size_t std::aligned_union< _Len, _Types >::alignment_value
- static constexpr _Tp **std::integral_constant< _Tp, __v >::value**

- template<typename... _Tp>
  using std::common_reference_t
- template<typename _Tp, typename _Up>
  constexpr bool std::is_layout_compatible_v
- template<typename _Base, typename _Derived>
  constexpr bool std::is_pointer_interconvertible_base_of_v
- template<typename _S1, typename _S2, typename _M1, typename _M2>
  constexpr bool std::is_corresponding_member (_M1 _S1::*__m1, _M2 _S2::*__m2) noexcept
- template<typename _Tp, typename _Mem>
  constexpr bool std::is_pointer_interconvertible_with_class (_Mem _Tp::*__mp) noexcept

3.15.6.1 Detailed Description

Template utilities for compile-time introspection and modification, including type classification traits, type property inspection traits and type transformation traits.

Since

C++11

3.15.6.2 Typedef Documentation

add_lvalue_reference_t

```
template<typename _Tp>
using std::add_lvalue_reference_t
```

Alias template for add_lvalue_reference.

add_pointer_t

```
template<typename _Tp>
using std::add_pointer_t
```

Alias template for add_pointer.

add_rvalue_reference_t

```
template<typename _Tp>
using std::add_rvalue_reference_t
```

Alias template for add_rvalue_reference.

aligned_storage_t

```
template<size_t _Len, size_t _Align = __aligned_storage_default_alignment(_Len)>
using std::aligned_storage_t
```
Alias template for aligned_storage.

common_reference_t

```
template<typename... _Tp>
using std::common_reference_t
```

Since

C++20

common_type_t

```
template<typename... _Tp>
using std::common_type_t
```
Alias template for common_type.

conditional_t

```
template<bool _Cond, typename _Iftrue, typename _Iffalse>
using std::conditional_t
```
Alias template for conditional.

decay_t

```
template<typename _Tp>
using std::decay_t
```
Alias template for decay.

enable_if_t

```
template<bool _Cond, typename _Tp = void>
using std::enable_if_t
```
Alias template for enable_if.

false_type

```
using std::false_type
```
The type used as a compile-time boolean with false value.

make_signed_t

```
template<typename _Tp>
using std::make_signed_t
```
Alias template for make_signed.

make_unsigned_t

```
template<typename _Tp>
using std::make_unsigned_t
```
Alias template for make_unsigned.

remove_all_extents_t

```
template<typename _Tp>
using std::remove_all_extents_t
```
Alias template for remove_all_extents.

remove_extent_t

```
template<typename _Tp>
using std::remove_extent_t
```
Alias template for remove_extent.

remove_pointer_t

```
template<typename _Tp>
using std::remove_pointer_t
```
Alias template for remove_pointer.

remove_reference_t

```
template<typename _Tp>
using std::remove_reference_t
```
Alias template for remove_reference.

result_of_t

```
template<typename _Tp>
using std::result_of_t
```
Alias template for result_of.

true_type

```
using std::true_type
```
The type used as a compile-time boolean with true value.

type

```
template<size_t _Len, typename...  _Types>
using std::aligned_union< _Len, _Types >::type
```
The storage.

underlying_type_t

```
template<typename _Tp>
using std::underlying_type_t
```
Alias template for underlying_type.

3.15.6.3 Function Documentation

is_corresponding_member()

```
template<typename _S1, typename _S2, typename _M1, typename _M2>
bool std::is_corresponding_member (
             _M1 _S1::* __m1,
             _M2 _S2::* __m2)  [constexpr], [noexcept]
```

Since

C++20

is_pointer_interconvertible_with_class()

```
template<typename _Tp, typename _Mem>
bool std::is_pointer_interconvertible_with_class (
            _Mem _Tp::* __mp)  [constexpr], [noexcept]
```

True if __mp points to the first member of a standard-layout type.

Returns

true if s.*__mp is pointer-interconvertible with s

Since

C++20

swap() [1/2]

```
template<typename _Tp>
_Require< __not_< __is_tuple_like< _Tp > >, is_move_constructible< _Tp >, is_move_assignable<
_Tp > > std::swap (
            _Tp & __a,
            _Tp & __b)  [inline], [constexpr], [noexcept]
```

Swaps two values.

Parameters

_↩ _a	A thing of arbitrary type.
_↩ _b	Another thing of arbitrary type.

Returns

Nothing.

swap() [2/2]

```
template<typename _Tp, size_t _Nm>
__enable_if_t< __is_swappable< _Tp >::value > std::swap (
            _Tp(&) __a[_Nm],
            _Tp(&) __b[_Nm])  [inline], [constexpr], [noexcept]
```

Swap the contents of two arrays.

3.15.6.4 Variable Documentation

alignment_value

```
template<size_t _Len, typename...  _Types>
const size_t std::aligned_union< _Len, _Types >::alignment_value  [static]
```

The value of the strictest alignment of _Types.

is_layout_compatible_v

```
template<typename _Tp, typename _Up>
bool std::is_layout_compatible_v  [constexpr]
```

Since

C++20

is_pointer_interconvertible_base_of_v

```
template<typename _Base, typename _Derived>
bool std::is_pointer_interconvertible_base_of_v  [constexpr]
```

Since

C++20

3.15.7 Rational Arithmetic

Collaboration diagram for Rational Arithmetic:

Files

- file ratio

Classes

- struct std::ratio< _Num, _Den >
- struct std::ratio_equal< _R1, _R2 >
- struct std::ratio_greater< _R1, _R2 >
- struct std::ratio_greater_equal< _R1, _R2 >
- struct std::ratio_less< _R1, _R2 >
- struct std::ratio_less_equal< _R1, _R2 >
- struct std::ratio_not_equal< _R1, _R2 >

Typedefs

- using **std::atto**
- using **std::centi**
- using **std::deca**
- using **std::deci**
- using **std::exa**
- using **std::femto**
- using **std::giga**

- using **std::hecto**
- using **std::kilo**
- using **std::mega**
- using **std::micro**
- using **std::milli**
- using **std::nano**
- using **std::peta**
- using **std::pico**
- template<typename _R1, typename _R2>
  using std::ratio_add
- template<typename _R1, typename _R2>
  using std::ratio_divide
- template<typename _R1, typename _R2>
  using std::ratio_multiply
- template<typename _R1, typename _R2>
  using std::ratio_subtract
- using **std::tera**

Variables

- template<typename _R1, typename _R2>
  constexpr bool **std::ratio_equal_v**
- template<typename _R1, typename _R2>
  constexpr bool **std::ratio_greater_equal_v**
- template<typename _R1, typename _R2>
  constexpr bool **std::ratio_greater_v**
- template<typename _R1, typename _R2>
  constexpr bool **std::ratio_less_equal_v**
- template<typename _R1, typename _R2>
  constexpr bool **std::ratio_less_v**
- template<typename _R1, typename _R2>
  constexpr bool **std::ratio_not_equal_v**

3.15.7.1 Detailed Description

Compile time representation of finite rational numbers.

3.15.7.2 Typedef Documentation

ratio_add

```
template<typename _R1, typename _R2>
using std::ratio_add
```

ratio_add

ratio_divide

```
template<typename _R1, typename _R2>
using std::ratio_divide
```

ratio_divide

ratio_multiply

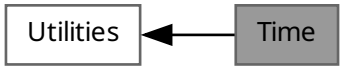
```
template<typename _R1, typename _R2>
using std::ratio_multiply
```
ratio_multiply

ratio_subtract

```
template<typename _R1, typename _R2>
using std::ratio_subtract
```
ratio_subtract

3.15.8 Time

Collaboration diagram for Time:



Files

- file chrono

Namespaces

- namespace std::chrono
- namespace std::literals::chrono_literals

Classes

- struct std::common_type< chrono::duration< _Rep, _Period > >
- struct std::common_type< chrono::duration< _Rep, _Period >, chrono::duration< _Rep, _Period > >
- struct std::common_type< chrono::duration< _Rep1, _Period1 >, chrono::duration< _Rep2, _Period2 > >
- struct std::common_type< chrono::time_point< _Clock, _Duration > >
- struct std::common_type< chrono::time_point< _Clock, _Duration >, chrono::time_point< _Clock, _Duration > >
- struct std::common_type< chrono::time_point< _Clock, _Duration1 >, chrono::time_point< _Clock, _Duration2 > >
- class std::chrono::duration< _Rep, _Period >
- struct std::chrono::duration_values< _Rep >
- class std::chrono::gps_clock
- class std::chrono::hh_mm_ss< _Duration >
- struct std::chrono::steady_clock
- struct std::chrono::system_clock
- class std::chrono::tai_clock
- class std::chrono::time_point< _Clock, _Dur >
- struct std::chrono::treat_as_floating_point< _Rep >
- class std::chrono::tzdb_list
- class std::chrono::utc_clock

Typedefs

- using std::chrono::days
- using **std::chrono::file_clock**
- template<typename _Duration>
  using **std::chrono::file_time**
- using **std::chrono::gps_seconds**
- template<typename _Duration>
  using **std::chrono::gps_time**
- using std::chrono::high_resolution_clock
- using std::chrono::hours
- using **std::chrono::local_days**
- using **std::chrono::local_seconds**
- template<typename _Duration>
  using **std::chrono::local_time**
- using std::chrono::microseconds
- using std::chrono::milliseconds
- using std::chrono::minutes
- using std::chrono::months
- using std::chrono::nanoseconds
- using std::chrono::seconds
- using **std::chrono::sys_days**
- using **std::chrono::sys_seconds**
- template<typename _Duration>
  using **std::chrono::sys_time**
- using **std::chrono::tai_seconds**
- template<typename _Duration>
  using **std::chrono::tai_time**
- using **std::chrono::utc_seconds**
- template<typename _Duration>
  using **std::chrono::utc_time**
- using std::chrono::weeks
- using std::chrono::years
- using **std::chrono::zoned_seconds**

Enumerations

- enum class **choose** { **earliest** , **latest** }

Functions

- constexpr **std::chrono::year_month_day::year_month_day** (const year_month_day_last &__ymdl) noexcept
- template<typename _Duration>
  void **std::chrono::__throw_bad_local_time** (const local_time< _Duration > &__tp, const local_info &__i)
- template<typename _Rep, typename _Period>
  constexpr enable_if_t< numeric_limits< _Rep >::is_signed, duration< _Rep, _Period > > std::chrono::abs (duration< _Rep, _Period > __d)
- template<typename _ToDur, typename _Rep, typename _Period>
  constexpr __enable_if_is_duration< _ToDur > std::chrono::ceil (const duration< _Rep, _Period > &__d)
- template<typename _ToDur, typename _Clock, typename _Dur>
  constexpr enable_if_t< __is_duration_v< _ToDur >, time_point< _Clock, _ToDur > > std::chrono::ceil (const time_point< _Clock, _Dur > &__tp)

- template<typename _DestClock, typename _SourceClock, typename _Duration>
  requires __detail::__clock_convs<_DestClock, _SourceClock, _Duration> || __detail::__clock_convs_sys<_DestClock, _SourceClock, _Duration> || __detail::__clock_convs_utc<_DestClock, _SourceClock, _Duration> || __detail::__clock_convs_sys_utc<_DestClock, _SourceClock, _Duration> || __detail::__clock_convs_utc_sys<_DestClock, _SourceClock, _Duration>
  auto std::chrono::clock_cast (const time_point< _SourceClock, _Duration > &__t)
- const time_zone * **std::chrono::current_zone** ()
- template<typename _ToDur, typename _Rep, typename _Period>
  constexpr __enable_if_is_duration< _ToDur > std::chrono::duration_cast (const duration< _Rep, _Period > &__d)
- template<typename _ToDur, typename _Rep, typename _Period>
  constexpr __enable_if_is_duration< _ToDur > std::chrono::floor (const duration< _Rep, _Period > &__d)
- template<typename _ToDur, typename _Clock, typename _Dur>
  constexpr enable_if_t< __is_duration_v< _ToDur >, time_point< _Clock, _ToDur > > std::chrono::floor (const time_point< _Clock, _Dur > &__tp)
- template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, day &__d, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Rep, typename _Period, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, duration< _Rep, _Period > &__d, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, file_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, gps_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, local_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, month &__m, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, month_day &__md, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, sys_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, tai_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, utc_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)

- template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, weekday &__wd, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, year &__y, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, year_month &__ym, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, year_month_day &__ymd, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _Duration>
  static utc_time< common_type_t< _Duration, seconds > > **std::chrono::utc_clock::from_sys** (const sys_time< _Duration > &__t)
- template<typename _Duration>
  leap_second_info **std::chrono::get_leap_second_info** (const utc_time< _Duration > &__ut)
- const tzdb & **std::chrono::get_tzdb** ()
- tzdb_list & **std::chrono::get_tzdb_list** ()
- constexpr bool **std::chrono::is_am** (const hours &__h) noexcept
- constexpr bool **std::chrono::is_pm** (const hours &__h) noexcept
- template<typename _Duration>
  __detail::__local_time_fmt< _Duration > std::chrono::local_time_format (local_time< _Duration > __time, const string *__abbrev=nullptr, const seconds *__offset_sec=nullptr)
- const time_zone * **std::chrono::locate_zone** (string_view __tz_name)
- constexpr hours **std::chrono::make12** (const hours &__h) noexcept
- constexpr hours **std::chrono::make24** (const hours &__h, bool __is_pm) noexcept
- constexpr bool **std::chrono::year_month_day::ok** () const noexcept
- constexpr chrono::day std::literals::chrono_literals::operator""d (unsigned long long __d) noexcept
- template<char... _Digits>
  constexpr chrono::hours std::literals::chrono_literals::operator""h ()
- constexpr chrono::duration< long double, ratio< 3600, 1 > > std::literals::chrono_literals::operator""h (long double __hours)
- template<char... _Digits>
  constexpr chrono::minutes std::literals::chrono_literals::operator""min ()
- constexpr chrono::duration< long double, ratio< 60, 1 > > std::literals::chrono_literals::operator""min (long double __mins)
- template<char... _Digits>
  constexpr chrono::milliseconds std::literals::chrono_literals::operator""ms ()
- constexpr chrono::duration< long double, milli > std::literals::chrono_literals::operator""ms (long double __msecs)
- template<char... _Digits>
  constexpr chrono::nanoseconds std::literals::chrono_literals::operator""ns ()
- constexpr chrono::duration< long double, nano > std::literals::chrono_literals::operator""ns (long double __nsecs)
- template<char... _Digits>
  constexpr chrono::seconds std::literals::chrono_literals::operator""s ()
- constexpr chrono::duration< long double > std::literals::chrono_literals::operator""s (long double __secs)
- template<char... _Digits>
  constexpr chrono::microseconds std::literals::chrono_literals::operator""us ()

- constexpr chrono::duration< long double, micro > std::literals::chrono_literals::operator""us (long double __usecs)
- constexpr chrono::year std::literals::chrono_literals::operator""y (unsigned long long __y) noexcept
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const day &__d)
- template<typename _CharT, typename _Traits, typename _Duration>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const file_time< _Duration > &__t)
- template<typename _CharT, typename _Traits, typename _Duration>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const gps_time< _Duration > &__t)
- template<typename _CharT, typename _Traits, typename _Duration>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const hh_mm_ss< _Duration > &__hms)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const local_info &__li)
- template<typename _CharT, typename _Traits, typename _Duration>
  requires requires(const sys_time<_Duration>& __st) { __os << __st; }
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const local_time< _Duration > &__lt)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const month &__m)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const month_day &__md)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const month_day_last &__mdl)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const month_weekday &__mwd)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const month_weekday_last &__mwdl)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const sys_days &__dp)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const sys_info &__i)
- template<typename _CharT, typename _Traits, typename _Duration>
  requires (!treat_as_floating_point_v<typename _Duration::rep>) && ratio_less_v<typename _Duration::period, days::period>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const sys_time< _Duration > &__tp)
- template<typename _CharT, typename _Traits, typename _Duration>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const tai_time< _Duration > &__t)
- template<typename _CharT, typename _Traits, typename _Duration>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const utc_time< _Duration > &__t)

- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const weekday &__wd)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const weekday_indexed &__wdi)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const weekday_last &__wdl)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const year &__y)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const year_month &__ym)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const year_month_day &__ymd)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const year_month_day_last &__ymdl)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const year_month_weekday &__ymwd)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const year_month_weekday_last &__ymwdl)
- template<typename _CharT, typename _Traits, typename _Duration, typename _TimeZonePtr>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const zoned_time< _Duration, _TimeZonePtr > &__t)
- template<typename _CharT, typename _Traits, typename _Rep, typename _Period>
  basic_ostream< _CharT, _Traits > & std::chrono::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const duration< _Rep, _Period > &__d)
- template<typename _Dur1, typename _TZPtr1, typename _Dur2, typename _TZPtr2>
  bool **std::chrono::operator==** (const zoned_time< _Dur1, _TZPtr1 > &__x, const zoned_time< _Dur2, _TZPtr2 > &__y)
- constexpr weekday_last **std::chrono::weekday::operator[ ]** (last_spec) const noexcept
- constexpr weekday_indexed **std::chrono::weekday::operator[ ]** (unsigned __index) const noexcept
- template<typename _CharT, __detail::__parsable< _CharT > _Parsable>
  auto **std::chrono::parse** (const _CharT *__fmt, _Parsable &__tp)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits, _Alloc>, __detail::__parsable< _CharT, _Traits, _StrT > _Parsable>
  auto **std::chrono::parse** (const _CharT *__fmt, _Parsable &__tp, basic_string< _CharT, _Traits, _Alloc > &_abbrev)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits, _Alloc>, __detail::__parsable< _CharT, _Traits, _StrT, minutes > _Parsable>
  auto **std::chrono::parse** (const _CharT *__fmt, _Parsable &__tp, basic_string< _CharT, _Traits, _Alloc > &_abbrev, minutes &__offset)
- template<typename _CharT, typename _Traits = char_traits<_CharT>, typename _StrT = basic_string<_CharT, _Traits>, __detail::__parsable< _CharT, _Traits, _StrT, minutes > _Parsable>
  auto **std::chrono::parse** (const _CharT *__fmt, _Parsable &__tp, minutes &__offset)
- template<typename _CharT, typename _Traits, typename _Alloc, __detail::__parsable< _CharT, _Traits > _Parsable>
  auto **std::chrono::parse** (const basic_string< _CharT, _Traits, _Alloc > &__fmt, _Parsable &__tp)

- template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits, _Alloc>, __detail::__parsable< _CharT, _Traits, _StrT > _Parsable>
  auto **std::chrono::parse** (const basic_string< _CharT, _Traits, _Alloc > &__fmt, _Parsable &__tp, basic_string< _CharT, _Traits, _Alloc > &__abbrev)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits, _Alloc>, __detail::__parsable< _CharT, _Traits, _StrT, minutes > _Parsable>
  auto **std::chrono::parse** (const basic_string< _CharT, _Traits, _Alloc > &__fmt, _Parsable &__tp, basic_string< _CharT, _Traits, _Alloc > &__abbrev, minutes &__offset)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits>, __detail::__parsable< _CharT, _Traits, _StrT, minutes > _Parsable>
  auto **std::chrono::parse** (const basic_string< _CharT, _Traits, _Alloc > &__fmt, _Parsable &__tp, minutes &__offset)
- const tzdb & **std::chrono::reload_tzdb** ()
- string **std::chrono::remote_version** ()
- template<typename _ToDur, typename _Rep, typename _Period>
  constexpr enable_if_t< __and_< __is_duration< _ToDur >, __not_< treat_as_floating_point< typename _ToDur::rep > > >::value, _ToDur > std::chrono::round (const duration< _Rep, _Period > &__d)
- template<typename _ToDur, typename _Clock, typename _Dur>
  constexpr enable_if_t< __is_duration_v< _ToDur > &&!treat_as_floating_point_v< typename _ToDur::rep >, time_point< _Clock, _ToDur > > std::chrono::round (const time_point< _Clock, _Dur > &__tp)
- template<typename _ToDur, typename _Clock, typename _Dur>
  constexpr __enable_if_t< __is_duration< _ToDur >::value, time_point< _Clock, _ToDur > > std::chrono::time_point_cast (const time_point< _Clock, _Dur > &__t)
- **std::chrono::zoned_time** () -> zoned_time< seconds >
- template<typename _TimeZonePtrOrName>
  std::chrono::zoned_time (_TimeZonePtrOrName &&) -> zoned_time< seconds, __time_zone_representation< _TimeZonePtrOrName > >
- template<typename _TimeZonePtrOrName, typename _Duration>
  std::chrono::zoned_time (_TimeZonePtrOrName &&, local_time< _Duration >, choose=choose::earliest) -> zoned_time< common_type_t< _Duration, seconds >, __time_zone_representation< _TimeZonePtrOrName > >
- template<typename _TimeZonePtrOrName, typename _Duration>
  std::chrono::zoned_time (_TimeZonePtrOrName &&, sys_time< _Duration >) -> zoned_time< common_type_t< _Duration, seconds >, __time_zone_representation< _TimeZonePtrOrName > >
- template<typename _Duration, typename _TimeZonePtrOrName, typename _TimeZonePtr2>
  std::chrono::zoned_time (_TimeZonePtrOrName &&, zoned_time< _Duration, _TimeZonePtr2 >, choose=choose::earliest) -> zoned_time< common_type_t< _Duration, seconds >, __time_zone_representation< _TimeZonePtrOrName > >
- template<typename _Duration>
  std::chrono::zoned_time (sys_time< _Duration >) -> zoned_time< common_type_t< _Duration, seconds > >

Variables

- template<typename _Tp, typename _Clock>
  constexpr bool **std::chrono::__is_time_point_for_v**
- template<typename _Clock, typename _Duration>
  constexpr bool **std::chrono::__is_time_point_for_v< time_point< _Clock, _Duration >, _Clock >**
- constexpr month **std::chrono::April**
- constexpr month **std::chrono::August**
- constexpr month **std::chrono::December**
- constexpr month **std::chrono::February**
- constexpr weekday **std::chrono::Friday**

- template<typename _Tp, typename = void>
  constexpr bool **std::chrono::is_clock_v**
- template<typename _Tp>
  constexpr bool **std::chrono::is_clock_v< _Tp, void_t< typename _Tp::rep, typename _Tp::period, typename _Tp::duration, typename _Tp::time_point::duration, decltype(_Tp::is_steady), decltype(_Tp::now())> >**
- template<> constexpr bool **std::chrono::is_clock_v< file_clock >**
- template<> constexpr bool **std::chrono::is_clock_v< gps_clock >**
- template<> constexpr bool **std::chrono::is_clock_v< steady_clock >**
- template<> constexpr bool **std::chrono::is_clock_v< system_clock >**
- template<> constexpr bool **std::chrono::is_clock_v< tai_clock >**
- template<> constexpr bool **std::chrono::is_clock_v< utc_clock >**
- constexpr month **std::chrono::January**
- constexpr month **std::chrono::July**
- constexpr month **std::chrono::June**
- constexpr last_spec **std::chrono::last**
- constexpr month **std::chrono::March**
- constexpr month **std::chrono::May**
- constexpr weekday **std::chrono::Monday**
- constexpr month **std::chrono::November**
- constexpr month **std::chrono::October**
- constexpr weekday **std::chrono::Saturday**
- constexpr month **std::chrono::September**
- constexpr weekday **std::chrono::Sunday**
- constexpr weekday **std::chrono::Thursday**
- template<typename _Rep>
  constexpr bool **std::chrono::treat_as_floating_point_v**
- template<> constexpr bool **std::chrono::treat_as_floating_point_v< double >**
- template<> constexpr bool **std::chrono::treat_as_floating_point_v< float >**
- template<> constexpr bool **std::chrono::treat_as_floating_point_v< int >**
- template<> constexpr bool **std::chrono::treat_as_floating_point_v< long >**
- template<> constexpr bool **std::chrono::treat_as_floating_point_v< long double >**
- template<> constexpr bool **std::chrono::treat_as_floating_point_v< long long >**
- constexpr weekday **std::chrono::Tuesday**
- constexpr weekday **std::chrono::Wednesday**

- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type std::chrono::operator- (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type operator+ (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)

- template<typename _Rep1, typename _Rep2, typename _Period>
  constexpr duration< __common_rep_t< _Rep2, _Rep1 >, _Period > std::chrono::operator* (const _Rep1 &__s, const duration< _Rep2, _Period > &__d)
- template<typename _Rep1, typename _Period, typename _Rep2>
  constexpr duration< __common_rep_t< _Rep1, __disable_if_is_duration< _Rep2 > >, _Period > std::chrono::operator/ (const duration< _Rep1, _Period > &__d, const _Rep2 &__s)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr common_type< _Rep1, _Rep2 >::type std::chrono::operator/ (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)

- template<typename _Rep1, typename _Period, typename _Rep2>
  constexpr duration< __common_rep_t< _Rep1, __disable_if_is_duration< _Rep2 > >, _Period > std::chrono::operator% (const duration< _Rep1, _Period > &__d, const _Rep2 &__s)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type std::chrono::operator% (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Period, typename _Rep2>
  constexpr duration< __common_rep_t< _Rep1, _Rep2 >, _Period > operator* (const duration< _Rep1, _Period > &__d, const _Rep2 &__s)

- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr bool std::chrono::operator< (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  requires three_way_comparable<common_type_t<_Rep1, _Rep2>>
  constexpr auto std::chrono::operator<=> (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr bool std::chrono::operator<= (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr bool std::chrono::operator> (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr bool std::chrono::operator>= (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr bool operator== (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)

- template<typename _Rep1, typename _Period1, typename _Clock, typename _Dur2>
  constexpr time_point< _Clock, typename common_type< duration< _Rep1, _Period1 >, _Dur2 >::type > std::chrono::operator+ (const duration< _Rep1, _Period1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)
- template<typename _Clock, typename _Dur1, typename _Rep2, typename _Period2>
  constexpr time_point< _Clock, typename common_type< _Dur1, duration< _Rep2, _Period2 > >::type > std::chrono::operator- (const time_point< _Clock, _Dur1 > &__lhs, const duration< _Rep2, _Period2 > &_rhs)
- template<typename _Clock, typename _Dur1, typename _Dur2>
  constexpr common_type< _Dur1, _Dur2 >::type std::chrono::operator- (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)
- template<typename _Clock, typename _Dur1, typename _Rep2, typename _Period2>
  constexpr time_point< _Clock, typename common_type< _Dur1, duration< _Rep2, _Period2 > >::type > operator+ (const time_point< _Clock, _Dur1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)

- template<typename _Clock, typename _Dur1, three_way_comparable_with< _Dur1 > _Dur2>
  constexpr auto **std::chrono::operator<=>** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)
- template<typename _Clock, typename _Dur1, typename _Dur2>
  constexpr bool **std::chrono::operator<** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)
- template<typename _Clock, typename _Dur1, typename _Dur2>
  constexpr bool **std::chrono::operator<=** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)

- template<typename _Clock, typename _Dur1, typename _Dur2>
  constexpr bool **std::chrono::operator>** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)
- template<typename _Clock, typename _Dur1, typename _Dur2>
  constexpr bool **std::chrono::operator>=** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)

3.15.8.1 Detailed Description

Classes and functions for time.

Since

C++11

3.15.8.2 Typedef Documentation

days

```
using std::chrono::days
```

days

high_resolution_clock

```
using std::chrono::high_resolution_clock
```

Highest-resolution clock.
This is the clock "with the shortest tick period." Alias to std::system_clock until higher-than-nanosecond definitions become feasible.

hours

```
using std::chrono::hours
```

hours

microseconds

```
using std::chrono::microseconds
```

microseconds

milliseconds

```
using std::chrono::milliseconds
```

milliseconds

minutes

```
using std::chrono::minutes
```

minutes

months

```
using std::chrono::months
```

months

nanoseconds

```
using std::chrono::nanoseconds
```

nanoseconds

seconds

```
using std::chrono::seconds
```
seconds

weeks

```
using std::chrono::weeks
```
weeks

years

```
using std::chrono::years
```
years

3.15.8.3 Function Documentation

abs()

```
template<typename _Rep, typename _Period>
enable_if_t< numeric_limits< _Rep >::is_signed, duration< _Rep, _Period > > std::chrono::abs (
            duration< _Rep, _Period > __d)  [nodiscard], [constexpr]
```
The absolute (non-negative) value of a duration.

Parameters

_↩ _d	A duration with a signed `rep` type.

Returns

A duration of the same type as the argument, with value |d|.

Since

C++17

ceil() [1/2]

```
template<typename _ToDur, typename _Rep, typename _Period>
__enable_if_is_duration< _ToDur > std::chrono::ceil (
            const duration< _Rep, _Period > & __d)  [nodiscard], [constexpr]
```
Convert a `duration` to type `ToDur` and round up.
If the duration cannot be represented exactly in the result type, returns the closest value that is greater than the argument.

Template Parameters

_ToDur	The result type must be a `duration`.

Parameters

_↩ _d	A duration.

Returns

The value of __d converted to type _ToDur.

Since

C++17

References duration_cast().
Referenced by ceil().

ceil() [2/2]

```
template<typename _ToDur, typename _Clock, typename _Dur>
enable_if_t< __is_duration_v< _ToDur >, time_point< _Clock, _ToDur > > std::chrono::ceil (
            const time_point< _Clock, _Dur > & __tp)  [nodiscard], [constexpr]
```

Convert a time_point to type ToDur and round up.
The result is the same time point as measured by the same clock, but using the specified duration to represent the time. If the time point cannot be represented exactly in the result type, returns the closest value that is greater than the argument.

Template Parameters

_ToDur	The duration type to use for the result.

Parameters

↩ _↩ ↩ _↩ *t*	A time point.

Returns

The value of __d converted to type _ToDur.

Since

C++17

References ceil().

clock_cast()

```
template<typename _DestClock, typename _SourceClock, typename _Duration>
requires __detail::__clock_convs<_DestClock, _SourceClock, _Duration> || __detail::__clock_convs↩
_sys<_DestClock, _SourceClock, _Duration> || __detail::__clock_convs_utc<_DestClock, _SourceClock,
_Duration> || __detail::__clock_convs_sys_utc<_DestClock, _SourceClock, _Duration> || __detail::_↩
_clock_convs_utc_sys<_DestClock, _SourceClock, _Duration>
auto std::chrono::clock_cast (
            const time_point< _SourceClock, _Duration > & __t)  [inline], [nodiscard]
```

Convert a time point to a different clock.

duration_cast()

```
template<typename _ToDur, typename _Rep, typename _Period>
__enable_if_is_duration< _ToDur > std::chrono::duration_cast (
            const duration< _Rep, _Period > & __d)  [nodiscard], [constexpr]
```

Convert a `duration` to type `ToDur`.
If the duration cannot be represented accurately in the result type, returns the result of integer truncation (i.e., rounded towards zero).

Template Parameters

_ToDur	The result type must be a `duration`.

Parameters

__d	A duration.

Returns

The value of `__d` converted to type `_ToDur`.

Since

C++11

Referenced by ceil(), floor(), std::this_thread::sleep_for(), and time_point_cast().

floor() [1/2]

```
template<typename _ToDur, typename _Rep, typename _Period>
__enable_if_is_duration< _ToDur > std::chrono::floor (
            const duration< _Rep, _Period > & __d)  [nodiscard], [constexpr]
```

Convert a `duration` to type `ToDur` and round down.
If the duration cannot be represented exactly in the result type, returns the closest value that is less than the argument.

Template Parameters

_ToDur	The result type must be a `duration`.

Parameters

__d	A duration.

Returns

The value of `__d` converted to type `_ToDur`.

Since

C++17

References duration_cast().
Referenced by floor(), and round().

floor() [2/2]

```
template<typename _ToDur, typename _Clock, typename _Dur>
enable_if_t< __is_duration_v< _ToDur >, time_point< _Clock, _ToDur > > std::chrono::floor (
            const time_point< _Clock, _Dur > & __tp)  [nodiscard], [constexpr]
```

Convert a `time_point` to type `ToDur` and round down.
The result is the same time point as measured by the same clock, but using the specified `duration` to represent the time. If the time point cannot be represented exactly in the result type, returns the closest value that is less than the argument.

Template Parameters

_ToDur	The `duration` type to use for the result.

Parameters

↩ _↩ ↩ _↩ *t*	A time point.

Returns

The value of `__d` converted to type `_ToDur`.

Since

C++17

References floor().

local_time_format()

```
template<typename _Duration>
__detail::__local_time_fmt< _Duration > std::chrono::local_time_format (
            local_time< _Duration > __time,
            const string * __abbrev = nullptr,
            const seconds * __offset_sec = nullptr)  [inline]
```

Return an object that asssociates timezone info with a local time.
A `chrono::local_time` object has no timezone associated with it. This function creates an object that allows formatting a `local_time` as though it refers to a timezone with the given abbreviated name and offset from UTC.

Since

C++20

operator"""d()

```
chrono::day std::literals::chrono_literals::operator""d (
            unsigned long long __d)  [constexpr], [noexcept]
```

Literal suffix for creating chrono::day objects.

Since

C++20

operator"""h() [1/2]

```
template<char... _Digits>
chrono::hours std::literals::chrono_literals::operator""h () [constexpr]
```

Literal suffix for durations of type `std::chrono::hours`

operator"""h() [2/2]

```
chrono::duration< long double, ratio< 3600, 1 > > std::literals::chrono_literals::operator""h (
            long double __hours) [constexpr]
```

Literal suffix for durations representing non-integer hours.

operator"""min() [1/2]

```
template<char... _Digits>
chrono::minutes std::literals::chrono_literals::operator""min () [constexpr]
```

Literal suffix for durations of type `std::chrono::minutes`

operator"""min() [2/2]

```
chrono::duration< long double, ratio< 60, 1 > > std::literals::chrono_literals::operator""min (
            long double __mins) [constexpr]
```

Literal suffix for durations representing non-integer minutes.

operator"""ms() [1/2]

```
template<char... _Digits>
chrono::milliseconds std::literals::chrono_literals::operator""ms () [constexpr]
```

Literal suffix for durations of type `std::chrono::milliseconds`

operator"""ms() [2/2]

```
chrono::duration< long double, milli > std::literals::chrono_literals::operator""ms (
            long double __msecs) [constexpr]
```

Literal suffix for durations representing non-integer milliseconds.

operator"""ns() [1/2]

```
template<char... _Digits>
chrono::nanoseconds std::literals::chrono_literals::operator""ns () [constexpr]
```

Literal suffix for durations of type `std::chrono::nanoseconds`

operator"""ns() [2/2]

```
chrono::duration< long double, nano > std::literals::chrono_literals::operator""ns (
            long double __nsecs) [constexpr]
```

Literal suffix for durations representing non-integer nanoseconds.

operator"""s() [1/2]

```
template<char... _Digits>
chrono::seconds std::literals::chrono_literals::operator""s () [constexpr]
```

Literal suffix for durations of type `std::chrono::seconds`

operator"""s() [2/2]

```
chrono::duration< long double > std::literals::chrono_literals::operator""s (
            long double __secs)  [constexpr]
```

Literal suffix for durations representing non-integer seconds.

operator"""us() [1/2]

```
template<char...  _Digits>
chrono::microseconds std::literals::chrono_literals::operator""us ()  [constexpr]
```

Literal suffix for durations of type `std::chrono::microseconds`

operator"""us() [2/2]

```
chrono::duration< long double, micro > std::literals::chrono_literals::operator""us (
            long double __usecs)  [constexpr]
```

Literal suffix for durations representing non-integer microseconds.

operator"""y()

```
chrono::year std::literals::chrono_literals::operator""y (
            unsigned long long __y)  [constexpr], [noexcept]
```

Literal suffix for creating chrono::year objects.

Since

C++20

operator%() [1/2]

```
template<typename _Rep1, typename _Period, typename _Rep2>
duration< __common_rep_t< _Rep1, __disable_if_is_duration< _Rep2 > >, _Period > std::chrono↩
::operator% (
            const duration< _Rep1, _Period > & __d,
            const _Rep2 & __s)  [constexpr]
```

Arithmetic operators for chrono::duration

operator%() [2/2]

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type std::chrono↩
::operator% (
            const duration< _Rep1, _Period1 > & __lhs,
            const duration< _Rep2, _Period2 > & __rhs)  [constexpr]
```

Arithmetic operators for chrono::duration

operator*() [1/2]

```
template<typename _Rep1, typename _Period, typename _Rep2>
duration< __common_rep_t< _Rep1, _Rep2 >, _Period > operator* (
            const duration< _Rep1, _Period > & __d,
            const _Rep2 & __s)  [related]
```

Arithmetic operators for chrono::duration

operator*() [2/2]

```
template<typename _Rep1, typename _Rep2, typename _Period>
duration< __common_rep_t< _Rep2, _Rep1 >, _Period > std::chrono::operator* (
            const _Rep1 & __s,
            const duration< _Rep2, _Period > & __d)  [constexpr]
```

Arithmetic operators for chrono::duration

operator+() [1/3]

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type operator+ (
            const duration< _Rep1, _Period1 > & __lhs,
            const duration< _Rep2, _Period2 > & __rhs)  [related]
```

The sum of two durations.

operator+() [2/3]

```
template<typename _Rep1, typename _Period1, typename _Clock, typename _Dur2>
time_point< _Clock, typename common_type< duration< _Rep1, _Period1 >, _Dur2 >::type > std↩
::chrono::operator+ (
            const duration< _Rep1, _Period1 > & __lhs,
            const time_point< _Clock, _Dur2 > & __rhs)  [constexpr]
```

Adjust a time point forwards by the given duration.

operator+() [3/3]

```
template<typename _Clock, typename _Dur1, typename _Rep2, typename _Period2>
time_point< _Clock, typename common_type< _Dur1, duration< _Rep2, _Period2 > >::type > operator+
(
            const time_point< _Clock, _Dur1 > & __lhs,
            const duration< _Rep2, _Period2 > & __rhs)  [related]
```

Adjust a time point forwards by the given duration.

operator-() [1/3]

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type std::chrono↩
::operator- (
            const duration< _Rep1, _Period1 > & __lhs,
            const duration< _Rep2, _Period2 > & __rhs)  [constexpr]
```

The difference between two durations.

operator-() [2/3]

```
template<typename _Clock, typename _Dur1, typename _Rep2, typename _Period2>
time_point< _Clock, typename common_type< _Dur1, duration< _Rep2, _Period2 > >::type > std↩
::chrono::operator- (
            const time_point< _Clock, _Dur1 > & __lhs,
            const duration< _Rep2, _Period2 > & __rhs)  [constexpr]
```

Adjust a time point backwards by the given duration.

operator-() [3/3]

```
template<typename _Clock, typename _Dur1, typename _Dur2>
common_type< _Dur1, _Dur2 >::type std::chrono::operator- (
```

```
            const time_point< _Clock, _Dur1 > & __lhs,
            const time_point< _Clock, _Dur2 > & __rhs)  [constexpr]
```

The difference between two time points (as a duration)

operator/() [1/2]

```
template<typename _Rep1, typename _Period, typename _Rep2>
duration< __common_rep_t< _Rep1, __disable_if_is_duration< _Rep2 > >, _Period > std::chrono↩
::operator/ (
            const duration< _Rep1, _Period > & __d,
            const _Rep2 & __s)  [constexpr]
```

Arithmetic operators for chrono::duration

operator/() [2/2]

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
common_type< _Rep1, _Rep2 >::type std::chrono::operator/ (
            const duration< _Rep1, _Period1 > & __lhs,
            const duration< _Rep2, _Period2 > & __rhs)  [constexpr]
```

Arithmetic operators for chrono::duration

operator<()

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
bool std::chrono::operator< (
            const duration< _Rep1, _Period1 > & __lhs,
            const duration< _Rep2, _Period2 > & __rhs)  [constexpr]
```

Comparisons for chrono::duration

operator<<() [1/3]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::chrono::operator<< (
            basic_ostream< _CharT, _Traits > & __os,
            const local_info & __li)
```

Writes a local_info object to an ostream in an unspecified format.

operator<<() [2/3]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::chrono::operator<< (
            basic_ostream< _CharT, _Traits > & __os,
            const sys_info & __i)
```

Writes a sys_info object to an ostream in an unspecified format.

operator<<() [3/3]

```
template<typename _CharT, typename _Traits, typename _Rep, typename _Period>
basic_ostream< _CharT, _Traits > & std::chrono::operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const duration< _Rep, _Period > & __d)  [inline]
```

Write a `chrono::duration` to an ostream.

Since

C++20

References std::ios_base::flags(), std::ios_base::getloc(), std::basic_ios< _CharT, _Traits >::imbue(), std::move(), operator<<(), and std::ios_base::precision().
Referenced by operator<<().

operator<=()

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
bool std::chrono::operator<= (
            const duration< _Rep1, _Period1 > & __lhs,
            const duration< _Rep2, _Period2 > & __rhs)  [constexpr]
```

Comparisons for chrono::duration

operator<=>()

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
requires three_way_comparable<common_type_t<_Rep1, _Rep2>>
auto std::chrono::operator<=> (
            const duration< _Rep1, _Period1 > & __lhs,
            const duration< _Rep2, _Period2 > & __rhs)  [constexpr]
```

Comparisons for chrono::duration

operator==()

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
bool operator== (
            const duration< _Rep1, _Period1 > & __lhs,
            const duration< _Rep2, _Period2 > & __rhs)  [related]
```

Comparisons for chrono::duration

operator>()

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
bool std::chrono::operator> (
            const duration< _Rep1, _Period1 > & __lhs,
            const duration< _Rep2, _Period2 > & __rhs)  [constexpr]
```

Comparisons for chrono::duration

operator>=()

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
bool std::chrono::operator>= (
            const duration< _Rep1, _Period1 > & __lhs,
            const duration< _Rep2, _Period2 > & __rhs)  [constexpr]
```

Comparisons for chrono::duration

round() [1/2]

```
template<typename _ToDur, typename _Rep, typename _Period>
enable_if_t< __and_< __is_duration< _ToDur >, __not_< treat_as_floating_point< typename _ToDur↩
::rep > > >::value, _ToDur > std::chrono::round (
            const duration< _Rep, _Period > & __d)  [nodiscard], [constexpr]
```

Convert a `duration` to type `ToDur` and round to the closest value.

If the duration cannot be represented exactly in the result type, returns the closest value, rounding ties to even.

Template Parameters

_ToDur	The result type must be a `duration` with a non-floating-point `rep` type.

Parameters

__d	A duration.

Returns

The value of `__d` converted to type `_ToDur`.

Since

C++17

References floor().
Referenced by round().

round() [2/2]

```
template<typename _ToDur, typename _Clock, typename _Dur>
enable_if_t< __is_duration_v< _ToDur > &&!treat_as_floating_point_v< typename _ToDur::rep >,
time_point< _Clock, _ToDur > > std::chrono::round (
            const time_point< _Clock, _Dur > & __tp)  [nodiscard], [constexpr]
```

Convert a `time_point` to type `ToDur` and round to the closest value.
The result is the same time point as measured by the same clock, but using the specified `duration` to represent the time. If the time point cannot be represented exactly in the result type, returns the closest value, rounding ties to even.

Template Parameters

_ToDur	The `duration` type to use for the result, which must have a non-floating-point `rep` type.

Parameters

__t	A time point.

Returns

The value of `__d` converted to type `_ToDur`.

Since

C++17

References round().

time_point_cast()

```
template<typename _ToDur, typename _Clock, typename _Dur>
__enable_if_t< __is_duration< _ToDur >::value, time_point< _Clock, _ToDur > > std::chrono↩
::time_point_cast (
            const time_point< _Clock, _Dur > & __t)  [nodiscard], [constexpr]
```

Convert a `time_point` to use `duration` type `ToDur`.
The result is the same time point as measured by the same clock, but using the specified `duration` to represent the time. If the time point cannot be represented accurately in the result type, returns the result of integer truncation (i.e., rounded towards zero).

Template Parameters

_ToDur	The `duration` type to use for the result.

Parameters

↩ _↩ ↩ _↩ *t*	A time point.

Returns

The value of `__t` converted to use type `_ToDur`.

Since

C++11

References duration_cast().

4 Namespace Documentation

4.1 __gnu_cxx Namespace Reference

Namespaces

- namespace __detail
- namespace typelist

Classes

- struct __alloc_traits
- struct __common_pool_policy
- class __mt_alloc
- class __mt_alloc_base
- struct __per_type_pool_policy
- class __pool
- class __pool< false >
- class __pool< true >
- class __pool_alloc

- class __pool_alloc_base
- struct __pool_base
- class __rc_string_base
- class __scoped_lock
- class __versa_string
- struct _Caster
- struct _Char_types
- class _ExtPtr_allocator
- struct _Invalid_type
- class _Pointer_adapter
- class _Relative_pointer_impl
- class _Relative_pointer_impl< const _Tp >
- class _Std_pointer_impl
- class _Temporary_buffer
- struct _Unqualified_type
- class allocator
- struct annotate_base
- class binary_compose
- class bitmap_allocator
- struct char_traits
- struct character
- struct condition_base
- struct constant_binary_fun
- struct constant_unary_fun
- struct constant_void_fun
- class debug_allocator
- class enc_filebuf
- struct encoding_char_traits
- class encoding_state
- struct equal_to
- struct forced_error
- class free_list
- class hash_map
- class hash_multimap
- class hash_multiset
- class hash_set
- struct limit_condition
- class malloc_allocator
- class new_allocator
- struct pair
- struct project1st
- struct project2nd
- struct random_condition
- struct rb_tree
- class recursive_init_error
- class rope
- struct select1st
- struct select2nd
- class slist
- class stdio_filebuf
- class stdio_sync_filebuf

- class subtractive_rng
- struct temporary_buffer
- class throw_allocator_base
- struct throw_allocator_limit
- struct throw_allocator_random
- struct throw_value_base
- struct throw_value_limit
- struct throw_value_random
- class unary_compose

Typedefs

- typedef void(* **__destroy_handler**) (void *)
- template<typename _Tp>
  using __int_traits
- typedef __versa_string< char, std::char_traits< char >, std::allocator< char >, __rc_string_base > **__rc_string**
- typedef __vstring **__sso_string**
- typedef __versa_string< char16_t, std::char_traits< char16_t >, std::allocator< char16_t >, __rc_string_base > **__u16rc_string**
- typedef __u16vstring **__u16sso_string**
- typedef __versa_string< char16_t > **__u16vstring**
- typedef __versa_string< char32_t, std::char_traits< char32_t >, std::allocator< char32_t >, __rc_string_base > **__u32rc_string**
- typedef __u32vstring **__u32sso_string**
- typedef __versa_string< char32_t > **__u32vstring**
- typedef __versa_string< char > **__vstring**
- typedef __versa_string< wchar_t, std::char_traits< wchar_t >, std::allocator< wchar_t >, __rc_string_base > **__wrc_string**
- typedef __wvstring **__wsso_string**
- typedef __versa_string< wchar_t > **__wvstring**
- typedef rope< char > **crope**
- typedef __SIZE_TYPE__ **size_t**
- typedef rope< wchar_t > **wrope**

Enumerations

- enum { **_S_num_primes** }
- enum **_Lock_policy** { **_S_single** , **_S_mutex** , **_S_atomic** }

Functions

- void **__atomic_add** (volatile _Atomic_word *__mem, int __val)
- void **__atomic_add_dispatch** (_Atomic_word *__mem, int __val)
- void **__atomic_add_single** (_Atomic_word *__mem, int __val)
- template<class _Tp>
  void **__aux_require_boolean_expr** (const _Tp &__t)
- template<typename _ToType, typename _FromType>
  _ToType **__const_pointer_cast** (_FromType *__arg)
- template<typename _ToType, typename _FromType>
  _ToType **__const_pointer_cast** (const _FromType &__arg)
- template<typename _InputIterator, typename _Size, typename _OutputIterator>
  std::pair< _InputIterator, _OutputIterator > **__copy_n** (_InputIterator __first, _Size __count, _OutputIterator _↩
  _result, std::input_iterator_tag)

- template<typename _RAIterator, typename _Size, typename _OutputIterator>
  std::pair< _RAIterator, _OutputIterator > **__copy_n** (_RAIterator __first, _Size __count, _OutputIterator __result, std::random_access_iterator_tag)
- template<typename _InputIterator, typename _Distance>
  void **__distance** (_InputIterator __first, _InputIterator __last, _Distance &__n, std::input_iterator_tag)
- template<typename _RandomAccessIterator, typename _Distance>
  void **__distance** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Distance &__n, std::random_access_iterator_tag
- template<typename _ToType, typename _FromType>
  _ToType **__dynamic_pointer_cast** (_FromType *__arg)
- template<typename _ToType, typename _FromType>
  _ToType **__dynamic_pointer_cast** (const _FromType &__arg)
- void **__error_type_must_be_a_signed_integer_type** ()
- void **__error_type_must_be_an_integer_type** ()
- void **__error_type_must_be_an_unsigned_integer_type** ()
- _Atomic_word **__exchange_and_add** (volatile _Atomic_word *__mem, int __val)
- _Atomic_word **__exchange_and_add_dispatch** (_Atomic_word *__mem, int __val)
- _Atomic_word **__exchange_and_add_single** (_Atomic_word *__mem, int __val)
- template<class _Concept>
  constexpr void **__function_requires** ()
- template<typename _Type>
  constexpr bool **__is_null_pointer** (_Type *__ptr)
- template<typename _Type>
  constexpr bool **__is_null_pointer** (_Type)
- constexpr bool **__is_null_pointer** (std::nullptr_t)
- bool **__is_single_threaded** () noexcept
- template<typename _InputIterator1, typename _InputIterator2>
  int **__lexicographical_compare_3way** (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)
- int **__lexicographical_compare_3way** (const char *__first1, const char *__last1, const char *__first2, const char *__last2)
- int **__lexicographical_compare_3way** (const unsigned char *__first1, const unsigned char *__last1, const unsigned char *__first2, const unsigned char *__last2)
- template<typename _Tp>
  const _Tp & __median (const _Tp &__a, const _Tp &__b, const _Tp &__c)
- template<typename _Tp, typename _Compare>
  const _Tp & __median (const _Tp &__a, const _Tp &__b, const _Tp &__c, _Compare __comp)
- crope::reference **__mutable_reference_at** (crope &__c, std::size_t __i)
- template<typename _Tp, typename _Integer>
  _Tp **__power** (_Tp __x, _Integer __n)
- template<typename _Tp, typename _Integer, typename _MonoidOperation>
  _Tp **__power** (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)
- template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator, typename _Distance>
  _RandomAccessIterator **__random_sample** (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out, _RandomNumberGenerator &__rand, const _Distance __n)
- template<typename _InputIterator, typename _RandomAccessIterator, typename _Distance>
  _RandomAccessIterator **__random_sample** (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out, const _Distance __n)
- template<typename _ToType, typename _FromType>
  _ToType **__reinterpret_pointer_cast** (_FromType *__arg)
- template<typename _ToType, typename _FromType>
  _ToType **__reinterpret_pointer_cast** (const _FromType &__arg)
- _Slist_node_base * **__slist_make_link** (_Slist_node_base *__prev_node, _Slist_node_base *__new_node)

- _Slist_node_base * **__slist_previous** (_Slist_node_base *__head, const _Slist_node_base *__node)
- const _Slist_node_base * **__slist_previous** (const _Slist_node_base *__head, const _Slist_node_base *__node)
- _Slist_node_base * **__slist_reverse** (_Slist_node_base *__node)
- std::size_t **__slist_size** (_Slist_node_base *__node)
- void **__slist_splice_after** (_Slist_node_base *__pos, _Slist_node_base *__before_first, _Slist_node_base *__before_last)
- void **__slist_splice_after** (_Slist_node_base *__pos, _Slist_node_base *__head)
- template<typename _ToType, typename _FromType>
  _ToType __static_pointer_cast (_FromType *__arg)
- template<typename _ToType, typename _FromType>
  _ToType __static_pointer_cast (const _FromType &__arg)
- size_t **__stl_hash_string** (const char *__s)
- unsigned long **__stl_next_prime** (unsigned long __n)
- template<typename _TRet, typename _Ret = _TRet, typename _CharT, typename... _Base>
  _Ret **__stoa** (_TRet(*__convf)(const _CharT *, _CharT **, _Base...), const char *__name, const _CharT *__str, std::size_t *__idx, _Base... __base)
- void **__throw_concurrence_lock_error** ()
- void **__throw_concurrence_unlock_error** ()
- void **__throw_forced_error** ()
- template<typename _String, typename _CharT = typename _String::value_type>
  _String **__to_xstring** (int(*__convf)(_CharT *, std::size_t, const _CharT *, __builtin_va_list), std::size_t __n, const _CharT *__fmt,...)
- template<typename _InputIter, typename _Size, typename _ForwardIter>
  std::pair< _InputIter, _ForwardIter > **__uninitialized_copy_n** (_InputIter __first, _Size __count, _ForwardIter __result)
- template<typename _InputIter, typename _Size, typename _ForwardIter>
  std::pair< _InputIter, _ForwardIter > **__uninitialized_copy_n** (_InputIter __first, _Size __count, _ForwardIter __result, std::input_iterator_tag)
- template<typename _RandomAccessIter, typename _Size, typename _ForwardIter>
  std::pair< _RandomAccessIter, _ForwardIter > **__uninitialized_copy_n** (_RandomAccessIter __first, _Size __count, _ForwardIter __result, std::random_access_iterator_tag)
- template<typename _InputIter, typename _Size, typename _ForwardIter, typename _Allocator>
  std::pair< _InputIter, _ForwardIter > **__uninitialized_copy_n_a** (_InputIter __first, _Size __count, _ForwardIter __result, _Allocator __alloc)
- template<typename _InputIter, typename _Size, typename _ForwardIter, typename _Tp>
  std::pair< _InputIter, _ForwardIter > **__uninitialized_copy_n_a** (_InputIter __first, _Size __count, _ForwardIter __result, std::allocator< _Tp >)
- void __verbose_terminate_handler ()
- std::size_t _Bit_scan_forward (std::size_t __num)
- template<typename _ForwardIterator, typename _Allocator>
  void **_Destroy_const** (_ForwardIterator __first, _ForwardIterator __last, _Allocator __alloc)
- template<typename _ForwardIterator, typename _Tp>
  void **_Destroy_const** (_ForwardIterator __first, _ForwardIterator __last, std::allocator< _Tp >)
- template<class _CharT, class _Traits>
  void **_Rope_fill** (std::basic_ostream< _CharT, _Traits > &__o, std::size_t __n)
- template<class _CharT>
  bool **_Rope_is_simple** (_CharT *)
- bool **_Rope_is_simple** (char *)
- bool **_Rope_is_simple** (wchar_t *)
- template<class _Rope_iterator>
  void **_Rope_rotate** (_Rope_iterator __first, _Rope_iterator __middle, _Rope_iterator __last)

- template<class _CharT>
  void **_S_cond_store_eos** (_CharT &)
- void **_S_cond_store_eos** (char &__c)
- void **_S_cond_store_eos** (wchar_t &__c)
- template<class _CharT>
  _CharT **_S_eos** (_CharT *)
- template<class _CharT>
  bool **_S_is_basic_char_type** (_CharT *)
- bool **_S_is_basic_char_type** (char *)
- bool **_S_is_basic_char_type** (wchar_t *)
- template<class _CharT>
  bool **_S_is_one_byte_char_type** (_CharT *)
- bool **_S_is_one_byte_char_type** (char *)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type airy_ai (_Tp __x)
- float airy_aif (float __x)
- long double airy_ail (long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type airy_bi (_Tp __x)
- float airy_bif (float __x)
- long double airy_bil (long double __x)
- template<class _Operation1, class _Operation2>
  unary_compose< _Operation1, _Operation2 > compose1 (const _Operation1 &__fn1, const _Operation2 &_↩
  _fn2)
- template<class _Operation1, class _Operation2, class _Operation3>
  binary_compose< _Operation1, _Operation2, _Operation3 > compose2 (const _Operation1 &__fn1, const _↩
  Operation2 &__fn2, const _Operation3 &__fn3)
- template<typename _Tpa, typename _Tpc, typename _Tp>
  __gnu_cxx::__promote_3< _Tpa, _Tpc, _Tp >::__type conf_hyperg (_Tpa __a, _Tpc __c, _Tp __x)
- float conf_hypergf (float __a, float __c, float __x)
- long double conf_hypergl (long double __a, long double __c, long double __x)
- template<class _Result>
  constant_void_fun< _Result > constant0 (const _Result &__val)
- template<class _Result>
  constant_unary_fun< _Result, _Result > constant1 (const _Result &__val)
- template<class _Result>
  constant_binary_fun< _Result, _Result, _Result > constant2 (const _Result &__val)
- template<typename _InputIterator, typename _Size, typename _OutputIterator>
  std::pair< _InputIterator, _OutputIterator > copy_n (_InputIterator __first, _Size __count, _OutputIterator __↩
  result)
- template<typename _InputIterator, typename _Tp, typename _Size>
  void **count** (_InputIterator __first, _InputIterator __last, const _Tp &__value, _Size &__n)
- template<typename _InputIterator, typename _Predicate, typename _Size>
  void **count_if** (_InputIterator __first, _InputIterator __last, _Predicate __pred, _Size &__n)
- template<typename _InputIterator, typename _Distance>
  void distance (_InputIterator __first, _InputIterator __last, _Distance &__n)
- template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp>
  __gnu_cxx::__promote_4< _Tpa, _Tpb, _Tpc, _Tp >::__type hyperg (_Tpa __a, _Tpb __b, _Tpc __c, _Tp __x)
- float hypergf (float __a, float __b, float __c, float __x)
- long double hypergl (long double __a, long double __b, long double __c, long double __x)
- template<class _Tp>
  _Tp identity_element (std::multiplies< _Tp >)

- template<class _Tp>
  _Tp identity_element (std::plus< _Tp >)
- template<typename _ForwardIterator, typename _Tp>
  constexpr void iota (_ForwardIterator __first, _ForwardIterator __last, _Tp __value)
- template<typename _RAIter>
  constexpr bool **is_heap** (_RAIter, _RAIter)
- template<typename _FIter>
  constexpr bool **is_sorted** (_FIter, _FIter)
- template<typename _InputIterator1, typename _InputIterator2>
  int lexicographical_compare_3way (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)
- template<class _Ret, class _Tp, class _Arg>
  std::const_mem_fun1_t< _Ret, _Tp, _Arg > **mem_fun1** (_Ret(_Tp::*__f)(_Arg) const)
- template<class _Ret, class _Tp, class _Arg>
  std::mem_fun1_t< _Ret, _Tp, _Arg > **mem_fun1** (_Ret(_Tp::*__f)(_Arg))
- template<class _Ret, class _Tp, class _Arg>
  std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg > **mem_fun1_ref** (_Ret(_Tp::*__f)(_Arg) const)
- template<class _Ret, class _Tp, class _Arg>
  std::mem_fun1_ref_t< _Ret, _Tp, _Arg > **mem_fun1_ref** (_Ret(_Tp::*__f)(_Arg))
- template<typename _Tp1, typename _Tp2>
  bool **operator!=** (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool operator!= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool operator!= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool operator!= (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _Tp>
  bool **operator!=** (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)
- template<typename _Tp>
  bool **operator!=** (const _Pointer_adapter< _Tp > &__lhs, int __rhs)
- template<typename _Tp1, typename _Tp2>
  bool **operator!=** (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)
- template<typename _Tp1, typename _Tp2>
  bool **operator!=** (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<class _CharT, class _Alloc>
  bool **operator!=** (const _Rope_char_ptr_proxy< _CharT, _Alloc > &__x, const _Rope_char_ptr_proxy< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **operator!=** (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **operator!=** (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)
- template<class _Key, class _Tp, class _HashFn, class _EqlKey, class _Alloc>
  bool **operator!=** (const hash_map< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm1, const hash_map< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm2)
- template<class _Key, class _Tp, class _HF, class _EqKey, class _Alloc>
  bool **operator!=** (const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm1, const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm2)
- template<class _Val, class _HashFcn, class _EqualKey, class _Alloc>
  bool **operator!=** (const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)

- template<class _Value, class _HashFcn, class _EqualKey, class _Alloc>
  bool **operator!=** (const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs2)
- template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All>
  bool **operator!=** (const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht1, const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht2)
- template<class _CharT, class _Alloc>
  bool **operator!=** (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)
- template<class _Tp, class _Alloc>
  bool **operator!=** (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)
- template<typename _Tp>
  bool **operator!=** (int __lhs, const _Pointer_adapter< _Tp > &__rhs)
- template<typename _Cond>
  throw_value_base< _Cond > **operator*** (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  __versa_string< _CharT, _Traits, _Alloc, _Base > **operator+** (__versa_string< _CharT, _Traits, _Alloc, _Base > &&__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &&__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  __versa_string< _CharT, _Traits, _Alloc, _Base > **operator+** (__versa_string< _CharT, _Traits, _Alloc, _Base > &&__lhs, _CharT __rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  __versa_string< _CharT, _Traits, _Alloc, _Base > **operator+** (__versa_string< _CharT, _Traits, _Alloc, _Base > &&__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  __versa_string< _CharT, _Traits, _Alloc, _Base > **operator+** (__versa_string< _CharT, _Traits, _Alloc, _Base > &&__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  __versa_string< _CharT, _Traits, _Alloc, _Base > **operator+** (_CharT __lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &&__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  __versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (_CharT __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  __versa_string< _CharT, _Traits, _Alloc, _Base > **operator+** (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &&__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  __versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, _CharT __rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  __versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  __versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  __versa_string< _CharT, _Traits, _Alloc, _Base > **operator+** (const _CharT *__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &&__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  __versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<class _CharT, class _Alloc>
  _Rope_const_iterator< _CharT, _Alloc > **operator+** (const _Rope_const_iterator< _CharT, _Alloc > &__x, std::ptrdiff_t __n)

- template<class _CharT, class _Alloc>
  _Rope_iterator< _CharT, _Alloc > **operator+** (const _Rope_iterator< _CharT, _Alloc > &__x, std::ptrdiff_t __n)
- template<class _CharT, class _Alloc>
  rope< _CharT, _Alloc > **operator+** (const rope< _CharT, _Alloc > &__left, _CharT __right)
- template<class _CharT, class _Alloc>
  rope< _CharT, _Alloc > **operator+** (const rope< _CharT, _Alloc > &__left, const _CharT *__right)
- template<class _CharT, class _Alloc>
  rope< _CharT, _Alloc > **operator+** (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &_↩
  _right)
- template<typename _Cond>
  throw_value_base< _Cond > **operator+** (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- template<class _CharT, class _Alloc>
  _Rope_const_iterator< _CharT, _Alloc > **operator+** (std::ptrdiff_t __n, const _Rope_const_iterator< _CharT, _Alloc > &__x)
- template<class _CharT, class _Alloc>
  _Rope_iterator< _CharT, _Alloc > **operator+** (std::ptrdiff_t __n, const _Rope_iterator< _CharT, _Alloc > &__x)
- template<class _CharT, class _Alloc>
  rope< _CharT, _Alloc > & **operator+=** (rope< _CharT, _Alloc > &__left, _CharT __right)
- template<class _CharT, class _Alloc>
  rope< _CharT, _Alloc > & **operator+=** (rope< _CharT, _Alloc > &__left, const _CharT *__right)
- template<class _CharT, class _Alloc>
  rope< _CharT, _Alloc > & **operator+=** (rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)
- template<class _CharT, class _Alloc>
  std::ptrdiff_t **operator-** (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _↩
  CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  _Rope_const_iterator< _CharT, _Alloc > **operator-** (const _Rope_const_iterator< _CharT, _Alloc > &__x, std↩
  ::ptrdiff_t __n)
- template<class _CharT, class _Alloc>
  std::ptrdiff_t **operator-** (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  _Rope_iterator< _CharT, _Alloc > **operator-** (const _Rope_iterator< _CharT, _Alloc > &__x, std::ptrdiff_t __n)
- template<typename _Cond>
  throw_value_base< _Cond > **operator-** (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- template<typename _Tp1, typename _Tp2>
  bool **operator**< (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool operator< (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool operator< (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool operator< (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _Tp1, typename _Tp2>
  bool **operator**< (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)
- template<typename _Tp1, typename _Tp2>
  bool **operator**< (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<class _CharT, class _Alloc>
  bool **operator**< (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)

- template<class _CharT, class _Alloc>
  bool **operator**< (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)
- template<typename _Value, typename _Int, typename _St>
  bool **operator**< (const character< _Value, _Int, _St > &lhs, const character< _Value, _Int, _St > &rhs)
- template<class _CharT, class _Alloc>
  bool **operator**< (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)
- template<class _Tp, class _Alloc>
  bool **operator**< (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)
- template<typename _Cond>
  bool **operator**< (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- template<class _CharT, class _Traits, class _Alloc>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__o, const rope< _CharT, _Alloc > &__r)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::beta_distribution< _RealType > &__x)
- template<typename _UIntType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::hypergeometric_distribution< _UIntType > &__x)
- template<size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__x)
- template<typename _UIntType, size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t __msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const ↩ __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::triangular_distribution< _RealType > &__x)
- template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::uniform_inside_sphere_distribution< _Dimen, _RealType > &__x)
- template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::uniform_on_sphere_distribution< _Dimen, _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::von_mises_distribution< _RealType > &__x)
- template<typename _CharT, typename _Traits, typename _StoreT>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const _Pointer_adapter< _StoreT > &__p)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const arcsine_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const hoyt_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const k_distribution< _RealType > &__x)

- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const logistic_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const nakagami_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const pareto_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const rice_distribution< _RealType > &__x)
- std::ostream & **operator**<< (std::ostream &os, const annotate_base &__b)
- template<typename _Tp1, typename _Tp2>
  bool **operator**<= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool operator<= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool operator<= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool operator<= (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _Tp>
  bool **operator**<= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)
- template<typename _Tp1, typename _Tp2>
  bool **operator**<= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)
- template<typename _Tp1, typename _Tp2>
  bool **operator**<= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<class _CharT, class _Alloc>
  bool **operator**<= (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **operator**<= (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **operator**<= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)
- template<class _Tp, class _Alloc>
  bool **operator**<= (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)
- template<typename _Tp1, typename _Tp2>
  bool **operator==** (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<size_t _Dimen, typename _RealType>
  bool **operator==** (const __gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__d1, const __gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__d2)
- template<typename _UIntType, size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t __msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4>
  bool **operator==** (const __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__lhs, const __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__rhs)
- template<typename _Tp, typename _Poolp>
  bool **operator==** (const __mt_alloc< _Tp, _Poolp > &, const __mt_alloc< _Tp, _Poolp > &)
- template<typename _Tp>
  bool **operator==** (const __pool_alloc< _Tp > &, const __pool_alloc< _Tp > &)

- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool operator== (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool operator== (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool operator== (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _Tp>
  bool operator== (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)
- template<typename _Tp>
  bool **operator==** (const _Pointer_adapter< _Tp > &__lhs, int __rhs)
- template<typename _Tp1, typename _Tp2>
  bool **operator==** (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)
- template<typename _Tp1, typename _Tp2>
  bool **operator==** (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<class _CharT, class _Alloc>
  bool **operator==** (const _Rope_char_ptr_proxy< _CharT, _Alloc > &__x, const _Rope_char_ptr_proxy< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **operator==** (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **operator==** (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)
- template<typename _Tp1, typename _Tp2>
  bool **operator==** (const bitmap_allocator< _Tp1 > &, const bitmap_allocator< _Tp2 > &) throw ()
- template<typename _Value, typename _Int, typename _St>
  bool **operator==** (const character< _Value, _Int, _St > &lhs, const character< _Value, _Int, _St > &rhs)
- template<class _Key, class _Tp, class _HashFn, class _EqlKey, class _Alloc>
  bool **operator==** (const hash_map< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm1, const hash_map< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm2)
- template<class _Key, class _Tp, class _HF, class _EqKey, class _Alloc>
  bool **operator==** (const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm1, const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm2)
- template<class _Val, class _HashFcn, class _EqualKey, class _Alloc>
  bool **operator==** (const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)
- template<class _Value, class _HashFcn, class _EqualKey, class _Alloc>
  bool **operator==** (const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs2)
- template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All>
  bool **operator==** (const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht1, const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht2)
- template<class _CharT, class _Alloc>
  bool **operator==** (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)
- template<class _Tp, class _Alloc>
  bool **operator==** (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)
- template<typename _Tp, typename _Cond>
  bool **operator==** (const throw_allocator_base< _Tp, _Cond > &, const throw_allocator_base< _Tp, _Cond > &)
- template<typename _Cond>
  bool **operator==** (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- template<typename _Tp>
  bool **operator==** (int __lhs, const _Pointer_adapter< _Tp > &__rhs)

- template<typename _Tp1, typename _Tp2>
  bool **operator**> (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool operator> (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool operator> (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool operator> (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _Tp>
  bool **operator**> (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)
- template<typename _Tp1, typename _Tp2>
  bool **operator**> (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)
- template<typename _Tp1, typename _Tp2>
  bool **operator**> (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<class _CharT, class _Alloc>
  bool **operator**> (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **operator**> (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **operator**> (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)
- template<class _Tp, class _Alloc>
  bool **operator**> (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)
- template<typename _Tp1, typename _Tp2>
  bool **operator**>**=** (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool operator>= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool operator>= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool operator>= (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _Tp>
  bool **operator**>**=** (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)
- template<typename _Tp1, typename _Tp2>
  bool **operator**>**=** (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)
- template<typename _Tp1, typename _Tp2>
  bool **operator**>**=** (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<class _CharT, class _Alloc>
  bool **operator**>**=** (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **operator**>**=** (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **operator**>**=** (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)
- template<class _Tp, class _Alloc>
  bool **operator**>**=** (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, __gnu_cxx::beta_distribution< _RealType > &__x)

- template<typename _UIntType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, __gnu_↩
  cxx::hypergeometric_distribution< _UIntType > &__x)
- template<size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, __gnu_↩
  cxx::normal_mv_distribution< _Dimen, _RealType > &__x)
- template<typename _UIntType, size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t __msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, __gnu_↩
  cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __↩
  msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, __gnu_↩
  cxx::triangular_distribution< _RealType > &__x)
- template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, __gnu_↩
  cxx::uniform_inside_sphere_distribution< _Dimen, _RealType > &__x)
- template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, __gnu_↩
  cxx::uniform_on_sphere_distribution< _Dimen, _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, __gnu_↩
  cxx::von_mises_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, arcsine_↩
  distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, hoyt_↩
  distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, k_↩
  distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, logistic_↩
  distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, nakagami↩
  _distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, pareto_↩
  distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, rice_↩
  distribution< _RealType > &__x)
- template<typename _Tp, typename _Integer>
  _Tp power (_Tp __x, _Integer __n)
- template<typename _Tp, typename _Integer, typename _MonoidOperation>
  _Tp power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)
- template<typename _InputIterator, typename _RandomAccessIterator>
  _RandomAccessIterator random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator ↩
  __out_first, _RandomAccessIterator __out_last)

- template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator>
  _RandomAccessIterator random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator ↩
  __out_first, _RandomAccessIterator __out_last, _RandomNumberGenerator &__rand)
- template<typename _ForwardIterator, typename _OutputIterator, typename _Distance>
  _OutputIterator random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const _Distance __n)
- template<typename _ForwardIterator, typename _OutputIterator, typename _Distance, typename _RandomNumberGenerator>
  _OutputIterator random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const _Distance __n, _RandomNumberGenerator &__rand)
- void **rotate** (_Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __first, _Rope_iterator< char, __↩
  STL_DEFAULT_ALLOCATOR(char)> __middle, _Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __last)
- double **stod** (const __vstring &__str, std::size_t *__idx=0)
- double **stod** (const __wvstring &__str, std::size_t *__idx=0)
- float **stof** (const __vstring &__str, std::size_t *__idx=0)
- float **stof** (const __wvstring &__str, std::size_t *__idx=0)
- int **stoi** (const __vstring &__str, std::size_t *__idx=0, int __base=10)
- int **stoi** (const __wvstring &__str, std::size_t *__idx=0, int __base=10)
- long **stol** (const __vstring &__str, std::size_t *__idx=0, int __base=10)
- long **stol** (const __wvstring &__str, std::size_t *__idx=0, int __base=10)
- long double **stold** (const __vstring &__str, std::size_t *__idx=0)
- long double **stold** (const __wvstring &__str, std::size_t *__idx=0)
- long long **stoll** (const __vstring &__str, std::size_t *__idx=0, int __base=10)
- long long **stoll** (const __wvstring &__str, std::size_t *__idx=0, int __base=10)
- unsigned long **stoul** (const __vstring &__str, std::size_t *__idx=0, int __base=10)
- unsigned long **stoul** (const __wvstring &__str, std::size_t *__idx=0, int __base=10)
- unsigned long long **stoull** (const __vstring &__str, std::size_t *__idx, int __base=10)
- unsigned long long **stoull** (const __wvstring &__str, std::size_t *__idx=0, int __base=10)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  void swap (__versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _Tp>
  void **swap** (_ExtPtr_allocator< _Tp > &__larg, _ExtPtr_allocator< _Tp > &__rarg)
- template<class _CharT, class __Alloc>
  void **swap** (_Rope_char_ref_proxy< _CharT, __Alloc > __a, _Rope_char_ref_proxy< _CharT, __Alloc > __b)
- template<class _Key, class _Tp, class _HashFn, class _EqlKey, class _Alloc>
  void **swap** (hash_map< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm1, hash_map< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm2)
- template<class _Key, class _Tp, class _HashFn, class _EqlKey, class _Alloc>
  void **swap** (hash_multimap< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm1, hash_multimap< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm2)
- template<class _Val, class _HashFcn, class _EqualKey, class _Alloc>
  void **swap** (hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)
- template<class _Val, class _HashFcn, class _EqualKey, class _Alloc>
  void **swap** (hash_set< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, hash_set< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)
- template<class _Val, class _Key, class _HF, class _Extract, class _EqKey, class _All>
  void **swap** (hashtable< _Val, _Key, _HF, _Extract, _EqKey, _All > &__ht1, hashtable< _Val, _Key, _HF, _Extract, _EqKey, _All > &__ht2)
- template<class _CharT, class _Alloc>
  void **swap** (rope< _CharT, _Alloc > &__x, rope< _CharT, _Alloc > &__y)

- template<class _Tp, class _Alloc>
  void **swap** (slist< _Tp, _Alloc > &__x, slist< _Tp, _Alloc > &__y)
- template<typename _Cond>
  void **swap** (throw_value_base< _Cond > &__a, throw_value_base< _Cond > &__b)
- __vstring **to_string** (double __val)
- __vstring **to_string** (float __val)
- __vstring **to_string** (int __val)
- __vstring **to_string** (long __val)
- __vstring **to_string** (long double __val)
- __vstring **to_string** (long long __val)
- __vstring **to_string** (unsigned __val)
- __vstring **to_string** (unsigned long __val)
- __vstring **to_string** (unsigned long long __val)
- __wvstring **to_wstring** (double __val)
- __wvstring **to_wstring** (float __val)
- __wvstring **to_wstring** (int __val)
- __wvstring **to_wstring** (long __val)
- __wvstring **to_wstring** (long double __val)
- __wvstring **to_wstring** (long long __val)
- __wvstring **to_wstring** (unsigned __val)
- __wvstring **to_wstring** (unsigned long __val)
- __wvstring **to_wstring** (unsigned long long __val)
- template<typename _InputIter, typename _Size, typename _ForwardIter>
  std::pair< _InputIter, _ForwardIter > uninitialized_copy_n (_InputIter __first, _Size __count, _ForwardIter __result)

Variables

- const _Lock_policy **__default_lock_policy**
- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__e**
- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__gamma_e**
- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__ln_10**
- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__ln_2**
- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__ln_3**
- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__log10_e**
- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__log2_e**
- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__one_div_e**
- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__one_div_pi**
- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__one_div_root_2**
- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__phi**

- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__pi**
- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__pi_half**
- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__pi_quarter**
- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__pi_third**
- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__root_2**
- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__root_3**
- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__root_5**
- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__root_7**
- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__root_pi_div_2**
- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__two_div_pi**
- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__two_div_root_pi**
- template<typename _Value>
  const int **__numeric_traits_floating**< **_Value** >**::__digits10**
- template<typename _Value>
  const bool **__numeric_traits_floating**< **_Value** >**::__is_signed**
- template<typename _Value>
  const int **__numeric_traits_floating**< **_Value** >**::__max_digits10**
- template<typename _Value>
  const int **__numeric_traits_floating**< **_Value** >**::__max_exponent10**
- template<typename _Value>
  const int **__numeric_traits_integer**< **_Value** >**::__digits**
- template<typename _Value>
  const bool **__numeric_traits_integer**< **_Value** >**::__is_signed**
- template<typename _Value>
  const _Value **__numeric_traits_integer**< **_Value** >**::__max**
- template<typename _Value>
  const _Value **__numeric_traits_integer**< **_Value** >**::__min**
- template<typename _Tp>
  _Atomic_word **__pool_alloc**< **_Tp** >**::_S_force_new**
- template<typename _CharT, typename _Traits, typename _Alloc>
  __rc_string_base< _CharT, _Traits, _Alloc >::_Rep_empty **__rc_string_base**< **_CharT, _Traits, _Alloc** >**::_S_empty_rep**
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  const __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type **__versa_string**< **_CharT, _Traits, _Alloc, _Base** >**::npos**
- template<typename _PrimeType>
  const _PrimeType **_Hashtable_prime_list**< **_PrimeType** >**::__stl_prime_list** [_S_num_primes]
- template<typename _Tp>
  std::size_t **bitmap_allocator**< **_Tp** >**::_S_block_size**
- template<typename _Tp>
  bitmap_allocator< _Tp >::_BPVector::size_type **bitmap_allocator**< **_Tp** >**::_S_last_dealloc_index**

- template<typename _Tp>
  bitmap_allocator< _Tp >::_BPVector **bitmap_allocator< _Tp >::_S_mem_blocks**
- template<typename _Tp>
  bitmap_allocator< _Tp >::__mutex_type **bitmap_allocator< _Tp >::_S_mut**
- template<class _CharT, class _Alloc>
  rope< _CharT, _Alloc > **identity_element** (_Rope_Concat_fn< _CharT, _Alloc >)
- template<class _CharT, class _Alloc>
  _CharT **rope< _CharT, _Alloc >::_S_empty_c_str** [1]
- template<class _CharT, class _Alloc>
  const unsigned long **rope< _CharT, _Alloc >::_S_min_len** [int(__detail::_S_max_rope_depth)+1]
- template<class _CharT, class _Alloc>
  const rope< _CharT, _Alloc >::size_type **rope< _CharT, _Alloc >::npos**

4.1.1 Detailed Description

GNU extensions for public use.

4.1.2 Typedef Documentation

__int_traits

```
template<typename _Tp>
using __gnu_cxx::__int_traits
```

Convenience alias for __numeric_traits<integer-type>.

4.1.3 Function Documentation

__static_pointer_cast() [1/2]

```
template<typename _ToType, typename _FromType>
_ToType __gnu_cxx::__static_pointer_cast (
            _FromType * __arg)  [inline]
```

Casting operations for cases where _FromType is a standard pointer. _ToType can be a standard or non-standard pointer.

__static_pointer_cast() [2/2]

```
template<typename _ToType, typename _FromType>
_ToType __gnu_cxx::__static_pointer_cast (
            const _FromType & __arg)  [inline]
```

Casting operations for cases where _FromType is not a standard pointer. _ToType can be a standard or non-standard pointer. Given that _FromType is not a pointer, it must have a get() method that returns the standard pointer equivalent of the address it points to, and must have an element_type typedef which names the type it points to.

_Bit_scan_forward()

```
std::size_t __gnu_cxx::_Bit_scan_forward (
            std::size_t __num)  [inline]
```

Generic Version of the bsf instruction.
Referenced by __gnu_cxx::bitmap_allocator< _Tp1 >::_M_allocate_single_object().

iota()

```
template<typename _ForwardIterator, typename _Tp>
void std::iota (
            _ForwardIterator __first,
```

```
            _ForwardIterator __last,
            _Tp __value)  [constexpr]
```

Create a range of sequentially increasing values.
For each element in the range [first,last) assigns `value` and increments `value` as if by `++value`.

Parameters

__first	Start of range.
__last	End of range.
__value	Starting value.

Returns

Nothing.

operator"!=() [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator!= (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)  [inline]
```

Test difference of two strings.

Parameters

__lhs	First string.
__rhs	Second string.

Returns

True if *__lhs.compare*(*__rhs*) != 0. False otherwise.

operator"!=() [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator!= (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
            const _CharT * __rhs)  [inline]
```

Test difference of string and C string.

Parameters

__lhs	String.
__rhs	C string.

Returns

True if *__lhs.compare*(*__rhs*) != 0. False otherwise.

operator"!=() [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator!= (
            const _CharT * __lhs,
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)  [inline]
```

Test difference of C string and string.

Parameters

__lhs	C string.
__rhs	String.

Returns

True if *__rhs.compare*(*__lhs*) != 0. False otherwise.

operator+() [1/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (
            _CharT __lhs,
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)
```

Concatenate character and string.

Parameters

__lhs	First string.
__rhs	Last string.

Returns

New string with *__lhs* followed by *__rhs*.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append(), __gnu_cxx::__versa_string< _CharT, _Traits, _Allo
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::reserve(), and __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Bas

operator+() [2/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
            _CharT __rhs)
```

Concatenate string and character.

Parameters

__lhs	First string.
__rhs	Last string.

Returns

New string with *__lhs* followed by *__rhs*.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append(), __gnu_cxx::__versa_string< _CharT, _Traits, _Allo
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::reserve(), and __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Bas

operator+() [3/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)
```

Concatenate two strings.

Parameters

__lhs	First string.
__rhs	Last string.

Returns

New string with value of *__lhs* followed by *__rhs*.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append(), __gnu_cxx::__versa_string< _CharT, _Traits, _Allo
and __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size().

operator+() [4/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
            const _CharT * __rhs)
```

Concatenate string and C string.

Parameters

__lhs	First string.
__rhs	Last string.

Returns

New string with *__lhs* followed by *__rhs*.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size().

operator+() [5/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (
            const _CharT * __lhs,
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)
```

Concatenate C string and string.

Parameters

__lhs	First string.
__rhs	Last string.

Returns

New string with value of *__lhs* followed by *__rhs*.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size().

operator<() [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator< (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)  [inline]
```

Test if string precedes string.

Parameters

__lhs	First string.
__rhs	Second string.

Returns

True if *__lhs* precedes *__rhs*. False otherwise.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare().

operator<() [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator< (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
            const _CharT * __rhs)  [inline]
```

Test if string precedes C string.

Parameters

__lhs	String.
__rhs	C string.

Returns

True if *__lhs* precedes *__rhs*. False otherwise.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare().

operator<() [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator< (
            const _CharT * __lhs,
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)  [inline]
```

Test if C string precedes string.

Parameters

__lhs	C string.
__rhs	String.

Returns

True if *__lhs* precedes *__rhs*. False otherwise.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare().

operator<=() [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator<= (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)  [inline]
```

Test if string doesn't follow string.

Parameters

__lhs	First string.
__rhs	Second string.

Returns

True if *__lhs* doesn't follow *__rhs*. False otherwise.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare().

operator<=() [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator<= (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
            const _CharT * __rhs)  [inline]
```

Test if string doesn't follow C string.

Parameters

__lhs	String.
__rhs	C string.

Returns

True if __*lhs* doesn't follow __*rhs*. False otherwise.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare().

operator<=() [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator<= (
            const _CharT * __lhs,
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)  [inline]
```

Test if C string doesn't follow string.

Parameters

__*lhs*	C string.
__*rhs*	String.

Returns

True if __*lhs* doesn't follow __*rhs*. False otherwise.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare().

operator==() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator== (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)  [inline]
```

Test equivalence of two strings.

Parameters

__*lhs*	First string.
__*rhs*	Second string.

Returns

True if __*lhs.compare*(__*rhs*) == 0. False otherwise.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::data(), and __gnu_cxx::__versa_string< _CharT, _Traits, _All

operator==() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator== (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
            const _CharT * __rhs)  [inline]
```

Test equivalence of string and C string.

Parameters

__lhs	String.
__rhs	C string.

Returns

True if *__lhs.compare*(*__rhs*) == 0. False otherwise.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::data(), and __gnu_cxx::__versa_string< _CharT, _Traits, _All

operator==() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator== (
            const _CharT * __lhs,
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)  [inline]
```

Test equivalence of C string and string.

Parameters

__lhs	C string.
__rhs	String.

Returns

True if *__rhs.compare*(*__lhs*) == 0. False otherwise.

operator==() [4/4]

```
template<typename _Tp>
bool __gnu_cxx::operator== (
            const _Pointer_adapter< _Tp > & __lhs,
            const _Pointer_adapter< _Tp > & __rhs)  [inline]
```

Comparison operators for _Pointer_adapter defer to the base class' comparison operators, when possible.

operator>() [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator> (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)  [inline]
```

Test if string follows string.

Parameters

__lhs	First string.
__rhs	Second string.

Returns

True if *__lhs* follows *__rhs*. False otherwise.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare().

operator>() [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator> (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
            const _CharT * __rhs)  [inline]
```

Test if string follows C string.

Parameters

__lhs	String.
__rhs	C string.

Returns

True if *__lhs* follows *__rhs*. False otherwise.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare().

operator>() [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator> (
            const _CharT * __lhs,
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)  [inline]
```

Test if C string follows string.

Parameters

__lhs	C string.
__rhs	String.

Returns

True if *__lhs* follows *__rhs*. False otherwise.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare().

operator>=() [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator>= (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)  [inline]
```

Test if string doesn't precede string.

Parameters

__lhs	First string.
__rhs	Second string.

Returns

True if *__lhs* doesn't precede *__rhs*. False otherwise.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare().

operator>=() [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator>= (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
            const _CharT * __rhs)  [inline]
```

Test if string doesn't precede C string.

Parameters

__lhs	String.
__rhs	C string.

Returns

True if *__lhs* doesn't precede *__rhs*. False otherwise.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare().

operator>=() [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator>= (
            const _CharT * __lhs,
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)  [inline]
```

Test if C string doesn't precede string.

Parameters

__lhs	C string.
__rhs	String.

Returns

True if *__lhs* doesn't precede *__rhs*. False otherwise.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare().

swap()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::swap (
            __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
            __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)  [inline]
```

Swap contents of two strings.

Parameters

__lhs	First string.
__rhs	Second string.

Exchanges the contents of *__lhs* and *__rhs* in constant time.
References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::swap().

4.2 __gnu_cxx::__detail Namespace Reference

Classes

- class __mini_vector
- class _Bitmap_counter
- class _Ffit_finder

Enumerations

- enum { **_S_max_rope_depth** }
- enum { **bits_per_byte** , **bits_per_block** }
- enum **_Tag** { **_S_leaf** , **_S_concat** , **_S_substringfn** , **_S_function** }

Functions

- void __bit_allocate (std::size_t *__pbmap, std::size_t __pos) throw ()
- void __bit_free (std::size_t *__pbmap, std::size_t __pos) throw ()
- template<typename _ForwardIterator, typename _Tp, typename _Compare>
  _ForwardIterator **__lower_bound** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)
- template<typename _AddrPair>
  std::size_t __num_bitmaps (_AddrPair __ap)
- template<typename _AddrPair>
  std::size_t __num_blocks (_AddrPair __ap)

4.2.1 Detailed Description

Implementation details not part of the namespace __gnu_cxx interface.

4.2.2 Function Documentation

__bit_allocate()

```
void __gnu_cxx::__detail::__bit_allocate (
            std::size_t * __pbmap,
            std::size_t __pos) throw ( )   [inline]
```

Mark a memory address as allocated by re-setting the corresponding bit in the bit-map.
Referenced by __gnu_cxx::bitmap_allocator< _Tp1 >::_M_allocate_single_object().

__bit_free()

```
void __gnu_cxx::__detail::__bit_free (
            std::size_t * __pbmap,
            std::size_t __pos) throw ( )   [inline]
```

Mark a memory address as free by setting the corresponding bit in the bit-map.
Referenced by __gnu_cxx::bitmap_allocator< _Tp1 >::_M_deallocate_single_object().

__num_bitmaps()

```
template<typename _AddrPair>
std::size_t __gnu_cxx::__detail::__num_bitmaps (
            _AddrPair __ap)  [inline]
```

The number of Bit-maps pointed to by the address pair passed to the function.
References __num_blocks().
Referenced by __gnu_cxx::bitmap_allocator< _Tp1 >::_M_allocate_single_object(), and __gnu_cxx::bitmap_allocator< _Tp1 >::_M_dea

__num_blocks()

```
template<typename _AddrPair>
std::size_t __gnu_cxx::__detail::__num_blocks (
            _AddrPair __ap)  [inline]
```

The number of Blocks pointed to by the address pair passed to the function.
Referenced by __num_bitmaps().

4.3 __gnu_cxx::typelist Namespace Reference

Functions

- template<typename Fn, typename Typelist>
  void **apply** (Fn &, Typelist)
- template<typename Fn, typename Typelist>
  void **apply_generator** (Fn &fn, Typelist)
- template<typename Fn, typename TypelistT, typename TypelistV>
  void **apply_generator** (Fn &fn, TypelistT, TypelistV)
- template<typename Gn, typename Typelist>
  void apply_generator (Gn &, Typelist)
- template<typename Gn, typename TypelistT, typename TypelistV>
  void **apply_generator** (Gn &, TypelistT, TypelistV)

4.3.1 Detailed Description

GNU typelist extensions for public compile-time use.

4.3.2 Function Documentation

apply_generator()

```
template<typename Gn, typename Typelist>
void __gnu_cxx::typelist::apply_generator (
            Gn & ,
            Typelist )
```

Apply all typelist types to generator functor.

4.4 __gnu_debug Namespace Reference

Classes

- class _After_nth_from
- struct _BeforeBeginHelper
- class _Equal_to
- class _Not_equal_to
- class _Safe_container
- class _Safe_forward_list

- class _Safe_iterator
- class _Safe_iterator_base
- class _Safe_local_iterator
- class _Safe_local_iterator_base
- class _Safe_node_sequence
- class _Safe_sequence
- class _Safe_sequence_base
- class _Safe_unordered_container
- class _Safe_unordered_container_base
- class _Safe_vector
- struct _Sequence_traits
- class basic_string
- class type_info

Typedefs

- typedef basic_string< char > **string**
- typedef basic_string< char16_t > u16string
- typedef basic_string< char32_t > u32string
- typedef basic_string< wchar_t > **wstring**

Enumerations

- enum **_Debug_msg_id** {
  __msg_valid_range , **__msg_insert_singular** , **__msg_insert_different** , **__msg_erase_bad** ,
  __msg_erase_different , **__msg_subscript_oob** , **__msg_empty** , **__msg_unpartitioned** ,
  __msg_unpartitioned_pred , **__msg_unsorted** , **__msg_unsorted_pred** , **__msg_not_heap** ,
  __msg_not_heap_pred , **__msg_bad_bitset_write** , **__msg_bad_bitset_read** , **__msg_bad_bitset_flip** ,
  __msg_self_splice , **__msg_splice_alloc** , **__msg_splice_bad** , **__msg_splice_other** ,
  __msg_splice_overlap , **__msg_init_singular** , **__msg_init_copy_singular** , **__msg_init_const_singular** ,
  __msg_copy_singular , **__msg_bad_deref** , **__msg_bad_inc** , **__msg_bad_dec** ,
  __msg_iter_subscript_oob , **__msg_advance_oob** , **__msg_retreat_oob** , **__msg_iter_compare_bad** ,
  __msg_compare_different , **__msg_iter_order_bad** , **__msg_order_different** , **__msg_distance_bad** ,
  __msg_distance_different , **__msg_deref_istream** , **__msg_inc_istream** , **__msg_output_ostream** ,
  __msg_deref_istreambuf , **__msg_inc_istreambuf** , **__msg_insert_after_end** , **__msg_erase_after_bad** ,
  __msg_valid_range2 , **__msg_local_iter_compare_bad** , **__msg_non_empty_range** , **__msg_self_move_assign** ,
  __msg_bucket_index_oob , **__msg_valid_load_factor** , **__msg_equal_allocs** , **__msg_insert_range_from_self** ,
  __msg_irreflexive_ordering }
- enum _Distance_precision {
  __dp_none , **__dp_equality** , **__dp_sign** , **__dp_sign_max_size** ,
  __dp_exact }

Functions

- template<typename _Iterator>
  constexpr _Iterator __base (_Iterator __it)
- template<typename _Iterator, typename _Sequence>
  _Iterator **__base** (const _Safe_iterator< _Iterator, _Sequence, std::random_access_iterator_tag > &__it)
- template<typename _Iterator>
  constexpr auto **__base** (const std::move_iterator< _Iterator > &__it) -> decltype(std::make_move_iterator(__base(__it.base())))

- template<typename _Iterator, typename _Sequence>
  constexpr std::reverse_iterator< _Iterator > **__base** (const std::reverse_iterator< _Safe_iterator< _Iterator, _↩
  Sequence, std::random_access_iterator_tag > > &__it)
- template<typename _InputIterator, typename _Size>
  constexpr bool **__can_advance** (_InputIterator, _Size)
- template<typename _InputIterator, typename _Diff>
  constexpr bool **__can_advance** (_InputIterator, const std::pair< _Diff, _Distance_precision > &, int)
- template<typename _Iterator, typename _Sequence, typename _Category, typename _Size>
  bool **__can_advance** (const _Safe_iterator< _Iterator, _Sequence, _Category > &, _Size)
- template<typename _Iterator, typename _Sequence, typename _Category, typename _Diff>
  bool **__can_advance** (const _Safe_iterator< _Iterator, _Sequence, _Category > &, const std::pair< _Diff, _Distance_precision > &, int)
- template<typename _Iterator, typename _Size>
  constexpr bool **__can_advance** (const std::move_iterator< _Iterator > &__it, _Size __n)
- template<typename _Iterator, typename _Diff>
  constexpr bool **__can_advance** (const std::move_iterator< _Iterator > &__it, const std::pair< _Diff, _Distance_precision > &__dist, int __way)
- template<typename _Iterator, typename _Size>
  constexpr bool **__can_advance** (const std::reverse_iterator< _Iterator > &__it, _Size __n)
- template<typename _Iterator, typename _Diff>
  constexpr bool **__can_advance** (const std::reverse_iterator< _Iterator > &__it, const std::pair< _Diff, _Distance_precision > &__dist, int __way)
- template<typename _ForwardIterator, typename _Tp>
  constexpr bool **__check_partitioned_lower** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__↩
  value)
- template<typename _ForwardIterator, typename _Tp, typename _Pred>
  constexpr bool **__check_partitioned_lower** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__↩
  value, _Pred __pred)
- template<typename _ForwardIterator, typename _Tp>
  constexpr bool **__check_partitioned_upper** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__↩
  value)
- template<typename _ForwardIterator, typename _Tp, typename _Pred>
  constexpr bool **__check_partitioned_upper** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__↩
  value, _Pred __pred)
- template<typename _Iterator>
  constexpr bool **__check_singular** (_Iterator const &)
- template<typename _Tp>
  constexpr bool __check_singular (_Tp *const &__ptr)
- bool __check_singular_aux (const _Safe_iterator_base *__x)
- bool **__check_singular_aux** (const class _Safe_iterator_base *)
- bool **__check_singular_aux** (const void *)
- template<typename _InputIterator>
  constexpr bool **__check_sorted** (const _InputIterator &__first, const _InputIterator &__last)
- template<typename _InputIterator, typename _Predicate>
  constexpr bool **__check_sorted** (const _InputIterator &__first, const _InputIterator &__last, _Predicate __pred)
- template<typename _ForwardIterator, typename _Predicate>
  constexpr bool **__check_sorted_aux** (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, std::forward_iterator_tag)
- template<typename _ForwardIterator>
  constexpr bool **__check_sorted_aux** (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- template<typename _InputIterator, typename _Predicate>
  constexpr bool **__check_sorted_aux** (const _InputIterator &, const _InputIterator &, _Predicate, std::input_iterator_tag)

- template<typename _InputIterator>
  constexpr bool **__check_sorted_aux** (const _InputIterator &, const _InputIterator &, std::input_iterator_tag)
- template<typename _InputIterator1, typename _InputIterator2>
  constexpr bool **__check_sorted_set** (const _InputIterator1 &__first, const _InputIterator1 &__last, const _Input↩ Iterator2 &)
- template<typename _InputIterator1, typename _InputIterator2, typename _Predicate>
  constexpr bool **__check_sorted_set** (const _InputIterator1 &__first, const _InputIterator1 &__last, const _Input↩ Iterator2 &, _Predicate __pred)
- template<typename _InputIterator, typename _Predicate>
  constexpr bool **__check_sorted_set_aux** (const _InputIterator &, const _InputIterator &, _Predicate, std::__↩ false_type)
- template<typename _InputIterator>
  constexpr bool **__check_sorted_set_aux** (const _InputIterator &, const _InputIterator &, std::__false_type)
- template<typename _InputIterator, typename _Predicate>
  constexpr bool **__check_sorted_set_aux** (const _InputIterator &__first, const _InputIterator &__last, _Predicate __pred, std::__true_type)
- template<typename _InputIterator>
  constexpr bool **__check_sorted_set_aux** (const _InputIterator &__first, const _InputIterator &__last, std::__↩ true_type)
- template<typename _CharT, typename _Integer>
  const _CharT * __check_string (const _CharT *__s, _Integer __n, const char *__file, unsigned int __line, const char *__function)
- template<typename _CharT>
  const _CharT * __check_string (const _CharT *__s, const char *__file, unsigned int __line, const char *__↩ function)
- template<typename _InputIterator>
  _InputIterator **__check_valid_range** (const _InputIterator &__first, const _InputIterator &__last, const char *_↩ _file, unsigned int __line, const char *__function)
- template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator>
  bool **__foreign_iterator** (const _Safe_iterator< _Iterator, _Sequence, _Category > &__it, _InputIterator __other, _InputIterator __other_end)
- template<typename _Iterator, typename _Sequence, typename _Category, typename _Integral>
  bool **__foreign_iterator_aux** (const _Safe_iterator< _Iterator, _Sequence, _Category > &, _Integral, _Integral, std::__true_type)
- template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator>
  bool **__foreign_iterator_aux** (const _Safe_iterator< _Iterator, _Sequence, _Category > &__it, _InputIterator __other, _InputIterator __other_end, std::__false_type)
- template<typename _Iterator, typename _Sequence, typename _Category, typename _OtherIterator, typename _OtherSequence, typename _OtherCategory>
  bool __foreign_iterator_aux2 (const _Safe_iterator< _Iterator, _Sequence, _Category > &, const _Safe_iterator< _OtherIterator, _OtherSequence, _OtherCategory > &, const _Safe_iterator< _OtherIterator, _OtherSequence, _OtherCategory > &)
- template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator>
  bool **__foreign_iterator_aux2** (const _Safe_iterator< _Iterator, _Sequence, _Category > &__it, const _Input↩ Iterator &__other, const _InputIterator &__other_end)
- template<typename _Iterator, typename _Sequence, typename _Category, typename _OtherIterator>
  bool __foreign_iterator_aux2 (const _Safe_iterator< _Iterator, _Sequence, _Category > &__it, const _Safe_iterator< _OtherIterator, _Sequence, _Category > &__other, const _Safe_iterator< _OtherIterator, _Sequence, _Category > &)
- template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator>
  bool **__foreign_iterator_aux3** (const _Safe_iterator< _Iterator, _Sequence, _Category > &, const _InputIterator &, const _InputIterator &, std::__false_type)

- template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator>
  bool **__foreign_iterator_aux3** (const _Safe_iterator< _Iterator, _Sequence, _Category > &__it, const _Input↩ Iterator &__other, const _InputIterator &__other_end, std::__true_type)
- template<typename _Iterator, typename _Sequence, typename _Category>
  bool **__foreign_iterator_aux4** (const _Safe_iterator< _Iterator, _Sequence, _Category > &,...)
- template<typename _Iterator, typename _Sequence, typename _Category>
  bool **__foreign_iterator_aux4** (const _Safe_iterator< _Iterator, _Sequence, _Category > &__it, const typename _Sequence::value_type *__other)
- template<typename _Iterator>
  constexpr _Distance_traits< _Iterator >::__type **__get_distance** (_Iterator __lhs, _Iterator __rhs)
- template<typename _Iterator>
  constexpr _Distance_traits< _Iterator >::__type **__get_distance** (_Iterator __lhs, _Iterator __rhs, std::input_iterator_tag)
- template<typename _Iterator>
  constexpr _Distance_traits< _Iterator >::__type __get_distance (_Iterator __lhs, _Iterator __rhs, std::random_access_iterator_tag)
- template<typename _Iterator>
  constexpr _Distance_traits< _Iterator >::__type **__get_distance** (const std::move_iterator< _Iterator > &__first, const std::move_iterator< _Iterator > &__last)
- template<typename _Iterator>
  constexpr _Distance_traits< _Iterator >::__type **__get_distance** (const std::reverse_iterator< _Iterator > &_↩ _first, const std::reverse_iterator< _Iterator > &__last)
- template<typename _Iterator>
  constexpr bool **__is_irreflexive** (_Iterator __it)
- template<typename _Iterator, typename _Pred>
  constexpr bool **__is_irreflexive_pred** (_Iterator __it, _Pred __pred)
- template<typename _Iterator>
  constexpr _Iterator **__unsafe** (_Iterator __it)
- template<typename _Iterator, typename _Sequence>
  _Iterator **__unsafe** (const _Safe_iterator< _Iterator, _Sequence > &__it)
- template<typename _Iterator, typename _UContainer>
  _Iterator **__unsafe** (const _Safe_local_iterator< _Iterator, _UContainer > &__it)
- template<typename _Iterator>
  constexpr auto **__unsafe** (const std::move_iterator< _Iterator > &__it) -> decltype(std::make_move_iterator(↩ __unsafe(__it.base())))
- template<typename _Iterator>
  constexpr auto **__unsafe** (const std::reverse_iterator< _Iterator > &__it) -> decltype(std::__make_reverse_↩ iterator(__unsafe(__it.base())))
- template<typename _InputIterator>
  constexpr bool **__valid_range** (_InputIterator __first, _InputIterator __last)
- template<typename _InputIterator>
  constexpr bool __valid_range (_InputIterator __first, _InputIterator __last, typename _Distance_traits< _Input↩ Iterator >::__type &__dist)
- template<typename _Iterator, typename _Sequence, typename _Category>
  bool **__valid_range** (const _Safe_iterator< _Iterator, _Sequence, _Category > &, const _Safe_iterator< _↩ Iterator, _Sequence, _Category > &)
- template<typename _Iterator, typename _Sequence, typename _Category>
  bool __valid_range (const _Safe_iterator< _Iterator, _Sequence, _Category > &, const _Safe_iterator< _Iterator, _Sequence, _Category > &, typename _Distance_traits< _Iterator >::__type &)
- template<typename _Iterator, typename _Sequence>
  bool **__valid_range** (const _Safe_local_iterator< _Iterator, _Sequence > &, const _Safe_local_iterator< _↩ Iterator, _Sequence > &)
- template<typename _Iterator, typename _Sequence>
  bool **__valid_range** (const _Safe_local_iterator< _Iterator, _Sequence > &, const _Safe_local_iterator< _↩ Iterator, _Sequence > &, typename _Distance_traits< _Iterator >::__type &)

- template<typename _Iterator, typename _UContainer>
  bool **__valid_range** (const _Safe_local_iterator< _Iterator, _UContainer > &__first, const _Safe_local_iterator< _Iterator, _UContainer > &__last)
- template<typename _Iterator, typename _UContainer>
  bool __valid_range (const _Safe_local_iterator< _Iterator, _UContainer > &__first, const _Safe_local_iterator< _Iterator, _UContainer > &__last, typename _Distance_traits< _Iterator >::__type &__dist_info)
- template<typename _Iterator>
  constexpr bool **__valid_range** (const std::move_iterator< _Iterator > &__first, const std::move_iterator< _Iterator > &__last, typename _Distance_traits< _Iterator >::__type &__dist)
- template<typename _Iterator>
  constexpr bool **__valid_range** (const std::reverse_iterator< _Iterator > &__first, const std::reverse_iterator< _Iterator > &__last, typename _Distance_traits< _Iterator >::__type &__dist)
- template<typename _InputIterator>
  constexpr bool __valid_range_aux (_InputIterator __first, _InputIterator __last, std::__false_type)
- template<typename _InputIterator>
  constexpr bool **__valid_range_aux** (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- template<typename _InputIterator>
  constexpr bool **__valid_range_aux** (_InputIterator __first, _InputIterator __last, std::random_access_iterator_tag)
- template<typename _InputIterator>
  constexpr bool **__valid_range_aux** (_InputIterator __first, _InputIterator __last, typename _Distance_traits< _InputIterator >::__type &__dist, std::__false_type)
- template<typename _Integral>
  constexpr bool __valid_range_aux (_Integral, _Integral, std::__true_type)
- template<typename _Integral>
  constexpr bool **__valid_range_aux** (_Integral, _Integral, typename _Distance_traits< _Integral >::__type &__dist, std::__true_type)
- template<typename _CharT, typename _Traits, typename _Allocator>
  std::basic_istream< _CharT, _Traits > & **getline** (std::basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Allocator > &__str)
- template<typename _CharT, typename _Traits, typename _Allocator>
  std::basic_istream< _CharT, _Traits > & **getline** (std::basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Allocator > &__str, _CharT __delim)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **operator!=** (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **operator!=** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **operator!=** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  basic_string< _CharT, _Traits, _Allocator > **operator+** (_CharT __lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  basic_string< _CharT, _Traits, _Allocator > **operator+** (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  basic_string< _CharT, _Traits, _Allocator > **operator+** (const basic_string< _CharT, _Traits, _Allocator > &_lhs, _CharT __rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  basic_string< _CharT, _Traits, _Allocator > **operator+** (const basic_string< _CharT, _Traits, _Allocator > &_lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  basic_string< _CharT, _Traits, _Allocator > **operator+** (const basic_string< _CharT, _Traits, _Allocator > &_lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)

- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **operator**< (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **operator**< (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **operator**< (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const basic_string< _CharT, _Traits, _Allocator > &__str)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **operator**<= (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **operator**<= (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **operator**<= (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **operator==** (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **operator==** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **operator==** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **operator**> (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **operator**> (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **operator**> (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **operator**>= (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **operator**>= (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **operator**>= (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Allocator > &__str)
- template<typename _CharT, typename _Traits, typename _Allocator>
  void **swap** (basic_string< _CharT, _Traits, _Allocator > &__lhs, basic_string< _CharT, _Traits, _Allocator > &↩
  __rhs)

4.4.1 Detailed Description

GNU debug classes for public use.

4.4.2 Typedef Documentation

u16string

```
typedef basic_string<char16_t> __gnu_debug::u16string
```

A string of `char16_t`.

u32string

```
typedef basic_string<char32_t> __gnu_debug::u32string
```
A string of `char32_t`.

4.4.3 Enumeration Type Documentation

_Distance_precision

```
enum __gnu_debug::_Distance_precision
```
The precision to which we can calculate the distance between two iterators.

4.4.4 Function Documentation

__base()

```
template<typename _Iterator>
_Iterator __gnu_debug::__base (
            _Iterator __it)  [inline], [constexpr]
```
Helper function to extract base iterator of random access safe iterator in order to reduce performance impact of debug mode. Limited to random access iterator because it is the only category for which it is possible to check for correct iterators order in the __valid_range function thanks to the < operator.
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_before_dereferenceable().

__check_singular()

```
template<typename _Tp>
bool __gnu_debug::__check_singular (
            _Tp *const & __ptr)  [inline], [constexpr]
```
Non-NULL pointers are nonsingular.

__check_singular_aux()

```
bool __gnu_debug::__check_singular_aux (
            const _Safe_iterator_base * __x)  [inline]
```
Iterators that derive from _Safe_iterator_base can be determined singular or non-singular.
References __gnu_debug::_Safe_iterator_base::_M_singular().

__check_string() [1/2]

```
template<typename _CharT, typename _Integer>
const _CharT * __gnu_debug::__check_string (
            const _CharT * __s,
            _Integer __n,
            const char * __file,
            unsigned int __line,
            const char * __function)  [inline]
```
Checks that __s is non-NULL or __n == 0, and then returns __s.

__check_string() [2/2]

```
template<typename _CharT>
const _CharT * __gnu_debug::__check_string (
            const _CharT * __s,
            const char * __file,
```

```
            unsigned int __line,
            const char * __function)  [inline]
```

Checks that __s is non-NULL and then returns __s.

__foreign_iterator_aux2() [1/2]

```
template<typename _Iterator, typename _Sequence, typename _Category, typename _OtherIterator,
typename _OtherSequence, typename _OtherCategory>
bool __gnu_debug::__foreign_iterator_aux2 (
            const _Safe_iterator< _Iterator, _Sequence, _Category > & ,
            const _Safe_iterator< _OtherIterator, _OtherSequence, _OtherCategory > & ,
            const _Safe_iterator< _OtherIterator, _OtherSequence, _OtherCategory > & )  [inline]
```

Handle debug iterators from different types of container.

__foreign_iterator_aux2() [2/2]

```
template<typename _Iterator, typename _Sequence, typename _Category, typename _OtherIterator>
bool __gnu_debug::__foreign_iterator_aux2 (
            const _Safe_iterator< _Iterator, _Sequence, _Category > & __it,
            const _Safe_iterator< _OtherIterator, _Sequence, _Category > & __other,
            const _Safe_iterator< _OtherIterator, _Sequence, _Category > & )  [inline]
```

Handle debug iterators from the same type of container.

__get_distance()

```
template<typename _Iterator>
_Distance_traits< _Iterator >::__type __gnu_debug::__get_distance (
            _Iterator __lhs,
            _Iterator __rhs,
            std::random_access_iterator_tag )  [inline], [constexpr]
```

Determine the distance between two iterators with some known precision.
References std::make_pair().

__valid_range() [1/3]

```
template<typename _InputIterator>
bool __gnu_debug::__valid_range (
            _InputIterator __first,
            _InputIterator __last,
            typename _Distance_traits< _InputIterator >::__type & __dist)  [inline], [constexpr]
```

Don't know what these iterators are, or if they are even iterators (we may get an integral type for InputIterator), so see if they are integral and pass them on to the next phase otherwise.
References __valid_range_aux().

__valid_range() [2/3]

```
template<typename _Iterator, typename _Sequence, typename _Category>
bool __gnu_debug::__valid_range (
            const _Safe_iterator< _Iterator, _Sequence, _Category > & __first,
            const _Safe_iterator< _Iterator, _Sequence, _Category > & __last,
            typename _Distance_traits< _Iterator >::__type & __dist)  [inline]
```

Safe iterators know how to check if they form a valid range.

__valid_range() [3/3]

```
template<typename _Iterator, typename _UContainer>
bool __gnu_debug::__valid_range (
             const _Safe_local_iterator< _Iterator, _UContainer > & __first,
             const _Safe_local_iterator< _Iterator, _UContainer > & __last,
             typename _Distance_traits< _Iterator >::__type & __dist_info)  [inline]
```

Safe local iterators know how to check if they form a valid range.

__valid_range_aux() [1/2]

```
template<typename _InputIterator>
bool __gnu_debug::__valid_range_aux (
             _InputIterator __first,
             _InputIterator __last,
             std::__false_type )  [inline], [constexpr]
```

We have iterators, so figure out what kind of iterators they are to see if we can check the range ahead of time.
References std::__iterator_category(), and __valid_range_aux().

__valid_range_aux() [2/2]

```
template<typename _Integral>
bool __gnu_debug::__valid_range_aux (
             _Integral ,
             _Integral ,
             std::__true_type )  [inline], [constexpr]
```

We say that integral types for a valid range, and defer to other routines to realize what to do with integral types instead of iterators.
Referenced by __valid_range(), and __valid_range_aux().

4.5 __gnu_internal Namespace Reference

4.5.1 Detailed Description

GNU implemenation details, not for public use or export. Used only when anonymous namespaces cannot be substituted.

4.6 __gnu_parallel Namespace Reference

Classes

- struct __accumulate_binop_reduct
- struct __accumulate_selector
- struct __adjacent_difference_selector
- struct __adjacent_find_selector
- class __binder1st
- class __binder2nd
- struct __count_if_selector
- struct __count_selector
- struct __fill_selector
- struct __find_first_of_selector
- struct __find_if_selector
- struct __for_each_selector
- struct __generate_selector
- struct __generic_find_selector

- struct __generic_for_each_selector
- struct __identity_selector
- struct __inner_product_selector
- struct __max_element_reduct
- struct __min_element_reduct
- struct __mismatch_selector
- struct __multiway_merge_3_variant_sentinel_switch
- struct __multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >
- struct __multiway_merge_4_variant_sentinel_switch
- struct __multiway_merge_4_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >
- struct __multiway_merge_k_variant_sentinel_switch
- struct __multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >
- struct __replace_if_selector
- struct __replace_selector
- struct __transform1_selector
- struct __transform2_selector
- class __unary_negate
- struct _DRandomShufflingGlobalData
- struct _DRSSorterPU
- struct _DummyReduct
- class _EqualFromLess
- struct _EqualTo
- class _GuardedIterator
- class _IteratorPair
- class _IteratorTriple
- struct _Job
- struct _Less
- class _Lexicographic
- class _LexicographicReverse
- class _LoserTree
- class _LoserTree< false, _Tp, _Compare >
- class _LoserTreeBase
- class _LoserTreePointer
- class _LoserTreePointer< false, _Tp, _Compare >
- class _LoserTreePointerBase
- class _LoserTreePointerUnguarded
- class _LoserTreePointerUnguarded< false, _Tp, _Compare >
- class _LoserTreePointerUnguardedBase
- struct _LoserTreeTraits
- class _LoserTreeUnguarded
- class _LoserTreeUnguarded< false, _Tp, _Compare >
- class _LoserTreeUnguardedBase
- struct _Multiplies
- struct _Nothing
- struct _Piece
- struct _Plus
- struct _PMWMSSortingData
- class _PseudoSequence
- class _PseudoSequenceIterator
- struct _QSBThreadLocal
- class _RandomNumber

- class _RestrictedBoundedConcurrentQueue
- struct _SamplingSorter
- struct _SamplingSorter< false, _RAIter, _StrictWeakOrdering >
- struct _Settings
- struct _SplitConsistently
- struct _SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIterator >
- struct _SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator >
- struct balanced_quicksort_tag
- struct balanced_tag
- struct constant_size_blocks_tag
- struct default_parallel_tag
- struct equal_split_tag
- struct exact_tag
- struct find_tag
- struct growing_blocks_tag
- struct multiway_mergesort_exact_tag
- struct multiway_mergesort_sampling_tag
- struct multiway_mergesort_tag
- struct omp_loop_static_tag
- struct omp_loop_tag
- struct parallel_tag
- struct quicksort_tag
- struct sampling_tag
- struct sequential_tag
- struct unbalanced_tag

Typedefs

- typedef unsigned short _BinIndex
- typedef int64_t _CASable
- typedef uint64_t _SequenceIndex
- typedef uint16_t _ThreadIndex

Enumerations

- enum _AlgorithmStrategy { **heuristic** , **force_sequential** , **force_parallel** }
- enum _FindAlgorithm { **GROWING_BLOCKS** , **CONSTANT_SIZE_BLOCKS** , **EQUAL_SPLIT** }
- enum _MultiwayMergeAlgorithm { **LOSER_TREE** }
- enum _Parallelism {
  sequential , parallel_unbalanced , parallel_balanced , parallel_omp_loop ,
  parallel_omp_loop_static , parallel_taskqueue }
- enum _PartialSumAlgorithm { **RECURSIVE** , **LINEAR** }
- enum _SortAlgorithm { **MWMS** , **QS** , **QS_BALANCED** }
- enum _SplittingAlgorithm { **SAMPLING** , **EXACT** }

Functions

- template<typename _Tp>
  _Tp **__add_omp** (volatile _Tp *__ptr, _Tp __addend)
- template<typename _RAIter, typename _DifferenceTp>
  void __calc_borders (_RAIter __elements, _DifferenceTp __length, _DifferenceTp *__off)
- template<typename _Tp>
  bool **__cas_omp** (volatile _Tp *__ptr, _Tp __comparand, _Tp __replacement)
- template<typename _Tp>
  bool __compare_and_swap (volatile _Tp *__ptr, _Tp __comparand, _Tp __replacement)
- template<typename _IIter, typename _OutputIterator>
  _OutputIterator **__copy_tail** (std::pair< _IIter, _IIter > __b, std::pair< _IIter, _IIter > __e, _OutputIterator __r)
- void __decode2 (_CASable __x, int &__a, int &__b)
- template<typename _RAIter, typename _DifferenceTp>
  void __determine_samples (_PMWMSSortingData< _RAIter > *__sd, _DifferenceTp __num_samples)
- _CASable __encode2 (int __a, int __b)
- template<typename _DifferenceType, typename _OutputIterator>
  _OutputIterator __equally_split (_DifferenceType __n, _ThreadIndex __num_threads, _OutputIterator __s)
- template<typename _DifferenceType>
  _DifferenceType __equally_split_point (_DifferenceType __n, _ThreadIndex __num_threads, _ThreadIndex __thread_no)
- template<typename _Tp>
  _Tp __fetch_and_add (volatile _Tp *__ptr, _Tp __addend)
- template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector>
  std::pair< _RAIter1, _RAIter2 > __find_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector)
- template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector>
  std::pair< _RAIter1, _RAIter2 > __find_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, constant_size_blocks_tag)
- template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector>
  std::pair< _RAIter1, _RAIter2 > __find_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, equal_split_tag)
- template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector>
  std::pair< _RAIter1, _RAIter2 > __find_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, growing_blocks_tag)
- template<typename _IIter, typename _UserOp, typename _Functionality, typename _Red, typename _Result>
  _UserOp __for_each_template_random_access (_IIter __begin, _IIter __end, _UserOp __user_op, _Functionality &__functionality, _Red __reduction, _Result __reduction_start, _Result &__output, typename std::iterator_traits< _IIter >::difference_type __bound, _Parallelism __parallelism_tag)
- template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>
  _Op __for_each_template_random_access_ed (_RAIter __begin, _RAIter __end, _Op __o, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator_traits< _RAIter >::difference_type __bound)
- template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>
  _Op __for_each_template_random_access_omp_loop (_RAIter __begin, _RAIter __end, _Op __o, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator_traits< _RAIter >::difference_type __bound)
- template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>
  _Op __for_each_template_random_access_omp_loop_static (_RAIter __begin, _RAIter __end, _Op __o, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator_traits< _RAIter >::difference_type __bound)
- template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>
  _Op __for_each_template_random_access_workstealing (_RAIter __begin, _RAIter __end, _Op __op, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator_traits< _RAIter >::difference_type __bound)

- _ThreadIndex **__get_max_threads** ()
- bool **__is_parallel** (const _Parallelism __p)
- template<typename _IIter, typename _Compare>
  bool __is_sorted (_IIter __begin, _IIter __end, _Compare __comp)
- template<typename _RAIter, typename _Compare>
  _RAIter __median_of_three_iterators (_RAIter __a, _RAIter __b, _RAIter __c, _Compare __comp)
- template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare>
  _OutputIterator __merge_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __↩
  end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)
- template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare>
  _OutputIterator __merge_advance_movc (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)
- template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare>
  _OutputIterator __merge_advance_usual (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)
- template<typename _RAIter1, typename _RAIter3, typename _Compare>
  _RAIter3 __parallel_merge_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter1 &__begin2, _RAIter1 _↩
  _end2, _RAIter3 __target, typename std::iterator_traits< _RAIter1 >::difference_type __max_length, _Compare __comp)
- template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _Compare>
  _RAIter3 __parallel_merge_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 _↩
  _end2, _RAIter3 __target, typename std::iterator_traits< _RAIter1 >::difference_type __max_length, _Compare __comp)
- template<typename _RAIter, typename _Compare>
  void __parallel_nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)
- template<typename _RAIter, typename _Compare>
  void __parallel_partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator __parallel_partial_sum (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __bin_op)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator __parallel_partial_sum_basecase (_IIter __begin, _IIter __end, _OutputIterator __result, _↩
  BinaryOperation __bin_op, typename std::iterator_traits< _IIter >::value_type __value)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator __parallel_partial_sum_linear (_IIter __begin, _IIter __end, _OutputIterator __result, _Binary↩
  Operation __bin_op, typename std::iterator_traits< _IIter >::difference_type __n)
- template<typename _RAIter, typename _Predicate>
  std::iterator_traits< _RAIter >::difference_type __parallel_partition (_RAIter __begin, _RAIter __end, _Predicate __pred, _ThreadIndex __num_threads)
- template<typename _RAIter, typename _RandomNumberGenerator>
  void __parallel_random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator __↩
  rng=_RandomNumber())
- template<typename _RAIter, typename _RandomNumberGenerator>
  void __parallel_random_shuffle_drs (_RAIter __begin, _RAIter __end, typename std::iterator_traits< _RAIter >↩
  ::difference_type __n, _ThreadIndex __num_threads, _RandomNumberGenerator &__rng)
- template<typename _RAIter, typename _RandomNumberGenerator>
  void __parallel_random_shuffle_drs_pu (_DRSSorterPU< _RAIter, _RandomNumberGenerator > *__pus)
- template<typename _IIter, typename _OutputIterator, typename _Compare>
  _OutputIterator **__parallel_set_difference** (_IIter __begin1, _IIter __end1, _IIter __begin2, _IIter __end2, _↩
  OutputIterator __result, _Compare __comp)
- template<typename _IIter, typename _OutputIterator, typename _Compare>
  _OutputIterator **__parallel_set_intersection** (_IIter __begin1, _IIter __end1, _IIter __begin2, _IIter __end2, _↩
  OutputIterator __result, _Compare __comp)

- template<typename _IIter, typename _OutputIterator, typename _Operation>
  _OutputIterator **__parallel_set_operation** (_IIter __begin1, _IIter __end1, _IIter __begin2, _IIter __end2, _OutputIterator __result, _Operation __op)
- template<typename _IIter, typename _OutputIterator, typename _Compare>
  _OutputIterator **__parallel_set_symmetric_difference** (_IIter __begin1, _IIter __end1, _IIter __begin2, _IIter __end2, _OutputIterator __result, _Compare __comp)
- template<typename _IIter, typename _OutputIterator, typename _Compare>
  _OutputIterator **__parallel_set_union** (_IIter __begin1, _IIter __end1, _IIter __begin2, _IIter __end2, _OutputIterator __result, _Compare __comp)
- template<bool __stable, typename _RAIter, typename _Compare, typename _Parallelism>
  void **__parallel_sort** (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)
- template<bool __stable, typename _RAIter, typename _Compare>
  void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, balanced_quicksort_tag __parallelism)
- template<bool __stable, typename _RAIter, typename _Compare>
  void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, default_parallel_tag __parallelism)
- template<bool __stable, typename _RAIter, typename _Compare>
  void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_exact_tag __parallelism)
- template<bool __stable, typename _RAIter, typename _Compare>
  void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_sampling_tag __parallelism)
- template<bool __stable, typename _RAIter, typename _Compare>
  void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_tag __parallelism)
- template<bool __stable, typename _RAIter, typename _Compare>
  void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, parallel_tag __parallelism)
- template<bool __stable, typename _RAIter, typename _Compare>
  void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, quicksort_tag __parallelism)
- template<typename _RAIter, typename _Compare>
  void __parallel_sort_qs (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)
- template<typename _RAIter, typename _Compare>
  void __parallel_sort_qs_conquer (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)
- template<typename _RAIter, typename _Compare>
  std::iterator_traits< _RAIter >::difference_type __parallel_sort_qs_divide (_RAIter __begin, _RAIter __end, _Compare __comp, typename std::iterator_traits< _RAIter >::difference_type __pivot_rank, typename std::iterator_traits< _RAIter >::difference_type __num_samples, _ThreadIndex __num_threads)
- template<typename _RAIter, typename _Compare>
  void __parallel_sort_qsb (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)
- template<typename _IIter, class _OutputIterator>
  _OutputIterator __parallel_unique_copy (_IIter __first, _IIter __last, _OutputIterator __result)
- template<typename _IIter, class _OutputIterator, class _BinaryPredicate>
  _OutputIterator __parallel_unique_copy (_IIter __first, _IIter __last, _OutputIterator __result, _BinaryPredicate __binary_pred)
- template<typename _RAIter, typename _Compare>
  void __qsb_conquer (_QSBThreadLocal< _RAIter > **__tls, _RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __iam, _ThreadIndex __num_threads, bool __parent_wait)
- template<typename _RAIter, typename _Compare>
  std::iterator_traits< _RAIter >::difference_type __qsb_divide (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)
- template<typename _RAIter, typename _Compare>
  void __qsb_local_sort_with_helping (_QSBThreadLocal< _RAIter > **__tls, _Compare &__comp, _ThreadIndex __iam, bool __wait)

- template<typename _RandomNumberGenerator>
  int __random_number_pow2 (int __logp, _RandomNumberGenerator &__rng)
- template<typename _Size>
  _Size __rd_log2 (_Size __n)
- template<typename _Tp>
  _Tp __round_up_to_pow2 (_Tp __x)
- template<typename __RAIter1, typename __RAIter2, typename _Pred>
  __RAIter1 __search_template (__RAIter1 __begin1, __RAIter1 __end1, __RAIter2 __begin2, __RAIter2 __end2, _Pred __pred)
- template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
  _RAIter3 __sequential_multiway_merge (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type↩ ::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)
- template<typename _RAIter, typename _RandomNumberGenerator>
  void __sequential_random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &__rng)
- template<typename _IIter>
  void __shrink (std::vector< _IIter > &__os_starts, size_t &__count_to_two, size_t &__range_length)
- template<typename _IIter>
  void __shrink_and_double (std::vector< _IIter > &__os_starts, size_t &__count_to_two, size_t &__range_length, const bool __make_twice)
- void __yield ()
- template<typename _IIter, typename _FunctorType>
  size_t list_partition (const _IIter __begin, const _IIter __end, _IIter *__starts, size_t *__lengths, const int __↩ num_parts, _FunctorType &__f, int __oversampling=0)
- template<typename _Tp>
  const _Tp & max (const _Tp &__a, const _Tp &__b)
- template<typename _Tp>
  const _Tp & min (const _Tp &__a, const _Tp &__b)
- template<typename _RanSeqs, typename _RankType, typename _RankIterator, typename _Compare>
  void multiseq_partition (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankIterator ↩ __begin_offsets, _Compare __comp=std::less< typename std::iterator_traits< typename std::iterator_traits< ↩ _RanSeqs >::value_type::first_type >::value_type >())
- template<typename _Tp, typename _RanSeqs, typename _RankType, typename _Compare>
  _Tp multiseq_selection (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankType &↩ __offset, _Compare __comp=std::less< _Tp >())
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut ↩ __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_tag __tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut ↩ __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sampling_tag __tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut ↩ __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut ↩ __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut ↩ __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))
- template<template< typename _RAI, typename _Cp > class iterator, typename _RAIterIterator, typename _RAIter3, typename _↩ DifferenceTp, typename _Compare>
  _RAIter3 multiway_merge_3_variant (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __↩ target, _DifferenceTp __length, _Compare __comp)

- template<template< typename _RAI, typename _Cp > class iterator, typename _RAIterIterator, typename _RAIter3, typename _↩ DifferenceTp, typename _Compare>
  _RAIter3 multiway_merge_4_variant (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __↩ target, _DifferenceTp __length, _Compare __comp)
- template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType>
  void multiway_merge_exact_splitting (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _Difference↩ Type __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _DifferenceType, _↩ DifferenceType > > *__pieces)
- template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
  _RAIter3 multiway_merge_loser_tree (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 _↩ _target, _DifferenceTp __length, _Compare __comp)
- template<typename _UnguardedLoserTree, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
  _RAIter3 multiway_merge_loser_tree_sentinel (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _↩ RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_↩ type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)
- template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
  _RAIter3 multiway_merge_loser_tree_unguarded (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, ↩ _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value↩ _type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)
- template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType>
  void multiway_merge_sampling_splitting (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _↩ DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _↩ DifferenceType, _DifferenceType > > *__pieces)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, ↩ _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_tag __tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _↩ RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, ↩ _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, ↩ _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, ↩ _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling_tag __tag)
- template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Splitter, type- name _Compare>
  _RAIter3 parallel_multiway_merge (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __↩ target, _Splitter __splitter, _DifferenceTp __length, _Compare __comp, _ThreadIndex __num_threads)
- template<bool __stable, bool __exact, typename _RAIter, typename _Compare>
  void parallel_sort_mwms (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)
- template<bool __stable, bool __exact, typename _RAIter, typename _Compare>
  void parallel_sort_mwms_pu (_PMWMSSortingData< _RAIter > *__sd, _Compare &__comp)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **stable_multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _↩ RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_tag __tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **stable_multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _↩ RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag)

- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **stable_multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **stable_multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **stable_multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling_tag __tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **stable_multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_tag __tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **stable_multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **stable_multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **stable_multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **stable_multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling_tag __tag)

Variables

- static const int _CASable_bits
- static const _CASable _CASable_mask

4.6.1 Detailed Description

GNU parallel code for public use.

4.6.2 Typedef Documentation

_BinIndex

```
typedef unsigned short __gnu_parallel::_BinIndex
```

Type to hold the index of a bin.
Since many variables of this type are allocated, it should be chosen as small as possible.

_CASable

```
typedef int64_t __gnu_parallel::_CASable
```

Longest compare-and-swappable integer type on this platform.

_SequenceIndex

```
typedef uint64_t __gnu_parallel::_SequenceIndex
```

Unsigned integer to index __elements. The total number of elements for each algorithm must fit into this type.

_ThreadIndex

```
typedef uint16_t __gnu_parallel::_ThreadIndex
```

Unsigned integer to index a thread number. The maximum thread number (for each processor) must fit into this type.

4.6.3 Enumeration Type Documentation

_AlgorithmStrategy

enum __gnu_parallel::_AlgorithmStrategy
Strategies for run-time algorithm selection:

_FindAlgorithm

enum __gnu_parallel::_FindAlgorithm
Find algorithms:

_MultiwayMergeAlgorithm

enum __gnu_parallel::_MultiwayMergeAlgorithm
Merging algorithms:

_Parallelism

enum __gnu_parallel::_Parallelism
Run-time equivalents for the compile-time tags.

Enumerator

sequential	Not parallel.
parallel_unbalanced	Parallel unbalanced (equal-sized chunks).
parallel_balanced	Parallel balanced (work-stealing).
parallel_omp_loop	Parallel with OpenMP dynamic load-balancing.
parallel_omp_loop_static	Parallel with OpenMP static load-balancing.
parallel_taskqueue	Parallel with OpenMP taskqueue construct.

_PartialSumAlgorithm

enum __gnu_parallel::_PartialSumAlgorithm
Partial sum algorithms: recursive, linear.

_SortAlgorithm

enum __gnu_parallel::_SortAlgorithm
Sorting algorithms:

_SplittingAlgorithm

enum __gnu_parallel::_SplittingAlgorithm
Sorting/merging algorithms: sampling, __exact.

4.6.4 Function Documentation

__calc_borders()

```
template<typename _RAIter, typename _DifferenceTp>
void __gnu_parallel::__calc_borders (
            _RAIter __elements,
```

```
            _DifferenceTp __length,
            _DifferenceTp * __off)
```

Precalculate __advances for Knuth-Morris-Pratt algorithm.

Parameters

__elements	Begin iterator of sequence to search for.
__length	Length of sequence to search for.
__off	Returned __offsets.

Referenced by __search_template().

__compare_and_swap()

```
template<typename _Tp>
bool __gnu_parallel::__compare_and_swap (
            volatile _Tp * __ptr,
            _Tp __comparand,
            _Tp __replacement)  [inline]
```

Compare-and-swap.
Compare `*__ptr` and `__comparand`. If equal, let `*__ptr=__replacement` and return `true`, return `false` otherwise.

Parameters

__ptr	Pointer to signed integer.
__comparand	Compare value.
__replacement	Replacement value.

Referenced by __parallel_partition(), __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::pop_back(), and __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::pop_front().

__decode2()

```
void __gnu_parallel::__decode2 (
            _CASable __x,
            int & __a,
            int & __b)  [inline]
```

Decode two integers from one gnu_parallel::_CASable.

Parameters

__x	__gnu_parallel::_CASable to decode integers from.
__a	First integer, to be decoded from the most-significant `_CASable_bits/2` bits of `__x`.
__b	Second integer, to be encoded in the least-significant `_CASable_bits/2` bits of `__x`.

See also

__encode2

References _CASable_bits, and _CASable_mask.
Referenced by __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::pop_back(), __gnu_parallel::_RestrictedBoundedConcurr
and __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::push_front().

__determine_samples()

```
template<typename _RAIter, typename _DifferenceTp>
void __gnu_parallel::__determine_samples (
            _PMWMSSortingData< _RAIter > * __sd,
            _DifferenceTp __num_samples)
```

Select _M_samples from a sequence.

Parameters

__sd	Pointer to algorithm data. _Result will be placed in __sd->_M_samples.
__num_samples	Number of _M_samples to select.

References __equally_split(), __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_samples, __gnu_parallel::_PMWMSSortingData<
and __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_starts.

__encode2()

```
_CASable __gnu_parallel::__encode2 (
            int __a,
            int __b)  [inline]
```

Encode two integers into one gnu_parallel::_CASable.

Parameters

_↩ _a	First integer, to be encoded in the most-significant _CASable_bits/2 bits.
_↩ _b	Second integer, to be encoded in the least-significant _CASable_bits/2 bits.

Returns

value encoding __a and __b.

See also

__decode2

References _CASable_bits.
Referenced by __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::_RestrictedBoundedConcurrentQueue(),
__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::pop_back(), __gnu_parallel::_RestrictedBoundedConcurrentQueue< _T
and __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::push_front().

__equally_split()

```
template<typename _DifferenceType, typename _OutputIterator>
_OutputIterator __gnu_parallel::__equally_split (
            _DifferenceType __n,
            _ThreadIndex __num_threads,
            _OutputIterator __s)
```

function to split a sequence into parts of almost equal size.
The resulting sequence __s of length __num_threads+1 contains the splitting positions when splitting the range [0,__n) into parts of almost equal size (plus minus 1). The first entry is 0, the last one n. There may result empty parts.

Parameters

__n	Number of elements
__num_threads	Number of parts
__s	Splitters

Returns

End of __splitter sequence, i.e. `__s+__num_threads+1`

Referenced by __determine_samples(), __find_template(), __parallel_partial_sum_linear(), __parallel_unique_copy(), __search_template(), and multiway_merge_exact_splitting().

__equally_split_point()

```
template<typename _DifferenceType>
_DifferenceType __gnu_parallel::__equally_split_point (
             _DifferenceType __n,
             _ThreadIndex __num_threads,
             _ThreadIndex __thread_no)
```

function to split a sequence into parts of almost equal size.
Returns the position of the splitting point between thread number __thread_no (included) and thread number __thread↩_no+1 (excluded).

Parameters

__n	Number of elements
__num_threads	Number of parts
__thread_no	Number of threads

Returns

splitting point

Referenced by __for_each_template_random_access_ed().

__fetch_and_add()

```
template<typename _Tp>
_Tp __gnu_parallel::__fetch_and_add (
             volatile _Tp * __ptr,
             _Tp __addend)  [inline]
```

Add a value to a variable, atomically.

Parameters

__ptr	Pointer to a signed integer.
__addend	Value to add.

Referenced by __find_template(), __for_each_template_random_access_workstealing(), __parallel_partition(), and __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::push_front().

__find_template() [1/4]

```
template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector>
std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_template (
            _RAIter1 __begin1,
            _RAIter1 __end1,
            _RAIter2 __begin2,
            _Pred __pred,
            _Selector __selector)  [inline]
```

Parallel std::find, switch for different algorithms.

Parameters

__begin1	Begin iterator of first sequence.
__end1	End iterator of first sequence.
__begin2	Begin iterator of second sequence. Must have same length as first sequence.
__pred	Find predicate.
__selector	_Functionality (e. g. std::find_if(), std::equal(),...)

Returns

Place of finding in both sequences.

References __find_template(), __gnu_parallel::_Settings::get(), and std::make_pair().
Referenced by __find_template().

__find_template() [2/4]

```
template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector>
std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_template (
            _RAIter1 __begin1,
            _RAIter1 __end1,
            _RAIter2 __begin2,
            _Pred __pred,
            _Selector __selector,
            constant_size_blocks_tag )
```

Parallel std::find, constant block size variant.

Parameters

__begin1	Begin iterator of first sequence.
__end1	End iterator of first sequence.
__begin2	Begin iterator of second sequence. Second __sequence must have same length as first sequence.
__pred	Find predicate.
__selector	_Functionality (e. g. std::find_if(), std::equal(),...)

Returns

Place of finding in both sequences.

See also

__gnu_parallel::_Settings::find_sequential_search_size

__gnu_parallel::_Settings::find_block_size There are two main differences between the growing blocks and the constant-size blocks variants.

1. For GB, the block size grows; for CSB, the block size is fixed.
2. For GB, the blocks are allocated dynamically; for CSB, the blocks are allocated in a predetermined manner, namely spacial round-robin.

References _GLIBCXX_CALL, __gnu_parallel::_Settings::find_initial_block_size, __gnu_parallel::_Settings::find_sequential_search_size, std::pair< _T1, _T2 >::first, __gnu_parallel::_Settings::get(), and std::min().

__find_template() [3/4]

```
template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector>
std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_template (
             _RAIter1 __begin1,
             _RAIter1 __end1,
             _RAIter2 __begin2,
             _Pred __pred,
             _Selector __selector,
             equal_split_tag )
```

Parallel std::find, equal splitting variant.

Parameters

__begin1	Begin iterator of first sequence.
__end1	End iterator of first sequence.
__begin2	Begin iterator of second sequence. Second __sequence must have same length as first sequence.
__pred	Find predicate.
__selector	_Functionality (e. g. std::find_if(), std::equal(),...)

Returns

Place of finding in both sequences.

References __equally_split(), and _GLIBCXX_CALL.

__find_template() [4/4]

```
template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector>
std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_template (
             _RAIter1 __begin1,
             _RAIter1 __end1,
             _RAIter2 __begin2,
             _Pred __pred,
             _Selector __selector,
             growing_blocks_tag )
```

Parallel std::find, growing block size variant.

Parameters

__begin1	Begin iterator of first sequence.

Parameters

__end1	End iterator of first sequence.
__begin2	Begin iterator of second sequence. Second __sequence must have same length as first sequence.
__pred	Find predicate.
__selector	_Functionality (e. g. std::find_if(), std::equal(),...)

Returns

Place of finding in both sequences.

See also

__gnu_parallel::_Settings::find_sequential_search_size

__gnu_parallel::_Settings::find_scale_factor

There are two main differences between the growing blocks and the constant-size blocks variants.

1. For GB, the block size grows; for CSB, the block size is fixed.
2. For GB, the blocks are allocated dynamically; for CSB, the blocks are allocated in a predetermined manner, namely spacial round-robin.

References __fetch_and_add(), _GLIBCXX_CALL, __gnu_parallel::_Settings::find_scale_factor, __gnu_parallel::_Settings::find_sequentia std::pair< _T1, _T2 >::first, __gnu_parallel::_Settings::get(), std::max(), and std::min().

__for_each_template_random_access()

```
template<typename _IIter, typename _UserOp, typename _Functionality, typename _Red, typename _↩
Result>
_UserOp __gnu_parallel::__for_each_template_random_access (
             _IIter __begin,
             _IIter __end,
             _UserOp __user_op,
             _Functionality & __functionality,
             _Red __reduction,
             _Result __reduction_start,
             _Result & __output,
             typename std::iterator_traits< _IIter >::difference_type __bound,
             _Parallelism __parallelism_tag)
```

Chose the desired algorithm by evaluating __parallelism_tag.

Parameters

__begin	Begin iterator of input sequence.
__end	End iterator of input sequence.
__user_op	A user-specified functor (comparator, predicate, associative operator,...)
__functionality	functor to *process* an element with __user_op (depends on desired functionality, e. g. accumulate, for_each,...
__reduction	Reduction functor.
__reduction_start	Initial value for reduction.
__output	Output iterator.
__bound	Maximum number of elements processed.

__parallelism_tag	Parallelization method

References __for_each_template_random_access_ed(), __for_each_template_random_access_omp_loop(), __for_each_template_rando parallel_omp_loop, parallel_omp_loop_static, and parallel_unbalanced.

__for_each_template_random_access_ed()

```
template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>
_Op __gnu_parallel::__for_each_template_random_access_ed (
              _RAIter __begin,
              _RAIter __end,
              _Op __o,
              _Fu & __f,
              _Red __r,
              _Result __base,
              _Result & __output,
              typename std::iterator_traits< _RAIter >::difference_type __bound)
```

Embarrassingly parallel algorithm for random access iterators, using hand-crafted parallelization by equal splitting the work.

Parameters

__begin	Begin iterator of element sequence.
__end	End iterator of element sequence.
__o	User-supplied functor (comparator, predicate, adding functor, ...)
__f	Functor to "process" an element with __op (depends on desired functionality, e. g. for std::for_each(), ...).
__r	Functor to "add" a single __result to the already processed elements (depends on functionality).
__base	Base value for reduction.
__output	Pointer to position where final result is written to
__bound	Maximum number of elements processed (e. g. for std::count_n()).

Returns

User-supplied functor (that may contain a part of the result).

References __equally_split_point(), and min().
Referenced by __for_each_template_random_access().

__for_each_template_random_access_omp_loop()

```
template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>
_Op __gnu_parallel::__for_each_template_random_access_omp_loop (
              _RAIter __begin,
              _RAIter __end,
              _Op __o,
              _Fu & __f,
              _Red __r,
              _Result __base,
              _Result & __output,
              typename std::iterator_traits< _RAIter >::difference_type __bound)
```

Embarrassingly parallel algorithm for random access iterators, using an OpenMP for loop.

Parameters

__begin	Begin iterator of element sequence.
__end	End iterator of element sequence.
__o	User-supplied functor (comparator, predicate, adding functor, etc.).
__f	Functor to *process* an element with __op (depends on desired functionality, e. g. for std::for_each(), ...).
__r	Functor to *add* a single __result to the already processed elements (depends on functionality).
__base	Base value for reduction.
__output	Pointer to position where final result is written to
__bound	Maximum number of elements processed (e. g. for std::count_n()).

Returns

User-supplied functor (that may contain a part of the result).

References min().
Referenced by __for_each_template_random_access().

__for_each_template_random_access_omp_loop_static()

```
template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>
_Op __gnu_parallel::__for_each_template_random_access_omp_loop_static (
            _RAIter __begin,
            _RAIter __end,
            _Op __o,
            _Fu & __f,
            _Red __r,
            _Result __base,
            _Result & __output,
            typename std::iterator_traits< _RAIter >::difference_type __bound)
```

Embarrassingly parallel algorithm for random access iterators, using an OpenMP for loop with static scheduling.

Parameters

__begin	Begin iterator of element sequence.
__end	End iterator of element sequence.
__o	User-supplied functor (comparator, predicate, adding functor, ...).
__f	Functor to *process* an element with __op (depends on desired functionality, e. g. for std::for_each(), ...).
__r	Functor to *add* a single __result to the already processed __elements (depends on functionality).
__base	Base value for reduction.
__output	Pointer to position where final result is written to
__bound	Maximum number of elements processed (e. g. for std::count_n()).

Returns

User-supplied functor (that may contain a part of the result).

References std::min().

__for_each_template_random_access_workstealing()

```
template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>
_Op __gnu_parallel::__for_each_template_random_access_workstealing (
              _RAIter __begin,
              _RAIter __end,
              _Op __op,
              _Fu & __f,
              _Red __r,
              _Result __base,
              _Result & __output,
              typename std::iterator_traits< _RAIter >::difference_type __bound)
```

Work stealing algorithm for random access iterators.
Uses O(1) additional memory. Synchronization at job lists is done with atomic operations.

Parameters

__begin	Begin iterator of element sequence.
__end	End iterator of element sequence.
__op	User-supplied functor (comparator, predicate, adding functor, ...).
__f	Functor to *process* an element with __op (depends on desired functionality, e. g. for std::for_each(), ...).
__r	Functor to *add* a single __result to the already processed elements (depends on functionality).
__base	Base value for reduction.
__output	Pointer to position where final result is written to
__bound	Maximum number of elements processed (e. g. for std::count_n()).

Returns

User-supplied functor (that may contain a part of the result).

References __fetch_and_add(), __yield(), _GLIBCXX_CALL, __gnu_parallel::_Job< _DifferenceTp >::_M_first, __gnu_parallel::_Job< _DifferenceTp >::_M_last, __gnu_parallel::_Job< _DifferenceTp >::_M_load, __gnu_parallel::_Settings::cache_l __gnu_parallel::_Settings::get(), max(), and min().
Referenced by __for_each_template_random_access().

__is_sorted()

```
template<typename _IIter, typename _Compare>
bool __gnu_parallel::__is_sorted (
              _IIter __begin,
              _IIter __end,
              _Compare __comp)
```

Check whether [__begin, __end) is sorted according to __comp.

Parameters

__begin	Begin iterator of sequence.
__end	End iterator of sequence.
__comp	Comparator.

Returns

`true` if sorted, `false` otherwise.

Referenced by __sequential_multiway_merge(), multiway_merge_loser_tree_sentinel(), and parallel_multiway_merge().

__median_of_three_iterators()

```
template<typename _RAIter, typename _Compare>
_RAIter __gnu_parallel::__median_of_three_iterators (
            _RAIter __a,
            _RAIter __b,
            _RAIter __c,
            _Compare __comp)
```

Compute the median of three referenced elements, according to __comp.

Parameters

__a	First iterator.
__b	Second iterator.
__c	Third iterator.
__comp	Comparator.

Referenced by __qsb_divide().

__merge_advance()

```
template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp,
typename _Compare>
_OutputIterator __gnu_parallel::__merge_advance (
            _RAIter1 & __begin1,
            _RAIter1 __end1,
            _RAIter2 & __begin2,
            _RAIter2 __end2,
            _OutputIterator __target,
            _DifferenceTp __max_length,
            _Compare __comp)  [inline]
```

Merge routine being able to merge only the __max_length smallest elements.
The __begin iterators are advanced accordingly, they might not reach __end, in contrast to the usual variant. Static switch on whether to use the conditional-move variant.

Parameters

__begin1	Begin iterator of first sequence.
__end1	End iterator of first sequence.
__begin2	Begin iterator of second sequence.
__end2	End iterator of second sequence.
__target	Target begin iterator.
__max_length	Maximum number of elements to merge.
__comp	Comparator.

Returns

Output end iterator.

References __merge_advance_movc(), and _GLIBCXX_CALL.
Referenced by __parallel_merge_advance(), and __sequential_multiway_merge().

__merge_advance_movc()

```
template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp,
typename _Compare>
_OutputIterator __gnu_parallel::__merge_advance_movc (
             _RAIter1 & __begin1,
             _RAIter1 __end1,
             _RAIter2 & __begin2,
             _RAIter2 __end2,
             _OutputIterator __target,
             _DifferenceTp __max_length,
             _Compare __comp)
```

Merge routine being able to merge only the __max_length smallest elements.
The __begin iterators are advanced accordingly, they might not reach __end, in contrast to the usual variant. Specially designed code should allow the compiler to generate conditional moves instead of branches.

Parameters

__begin1	Begin iterator of first sequence.
__end1	End iterator of first sequence.
__begin2	Begin iterator of second sequence.
__end2	End iterator of second sequence.
__target	Target begin iterator.
__max_length	Maximum number of elements to merge.
__comp	Comparator.

Returns

Output end iterator.

Referenced by __merge_advance().

__merge_advance_usual()

```
template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp,
typename _Compare>
_OutputIterator __gnu_parallel::__merge_advance_usual (
             _RAIter1 & __begin1,
             _RAIter1 __end1,
             _RAIter2 & __begin2,
             _RAIter2 __end2,
             _OutputIterator __target,
             _DifferenceTp __max_length,
             _Compare __comp)
```

Merge routine being able to merge only the __max_length smallest elements.
The __begin iterators are advanced accordingly, they might not reach __end, in contrast to the usual variant.

Parameters

__begin1	Begin iterator of first sequence.
__end1	End iterator of first sequence.
__begin2	Begin iterator of second sequence.
__end2	End iterator of second sequence.

Parameters

__target	Target begin iterator.
__max_length	Maximum number of elements to merge.
__comp	Comparator.

Returns

Output end iterator.

__parallel_merge_advance() [1/2]

```
template<typename _RAIter1, typename _RAIter3, typename _Compare>
_RAIter3 __gnu_parallel::__parallel_merge_advance (
            _RAIter1 & __begin1,
            _RAIter1 __end1,
            _RAIter1 & __begin2,
            _RAIter1 __end2,
            _RAIter3 __target,
            typename std::iterator_traits< _RAIter1 >::difference_type __max_length,
            _Compare __comp)  [inline]
```

Parallel merge routine being able to merge only the `__max_length` smallest elements.
The `__begin` iterators are advanced accordingly, they might not reach `__end`, in contrast to the usual variant. The functionality is projected onto parallel_multiway_merge.

Parameters

__begin1	Begin iterator of first sequence.
__end1	End iterator of first sequence.
__begin2	Begin iterator of second sequence.
__end2	End iterator of second sequence.
__target	Target begin iterator.
__max_length	Maximum number of elements to merge.
__comp	Comparator.

Returns

Output end iterator.

References std::make_pair(), multiway_merge_exact_splitting(), and parallel_multiway_merge().

__parallel_merge_advance() [2/2]

```
template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _Compare>
_RAIter3 __gnu_parallel::__parallel_merge_advance (
            _RAIter1 & __begin1,
            _RAIter1 __end1,
            _RAIter2 & __begin2,
            _RAIter2 __end2,
            _RAIter3 __target,
            typename std::iterator_traits< _RAIter1 >::difference_type __max_length,
            _Compare __comp)  [inline]
```

Merge routine fallback to sequential in case the iterators of the two input sequences are of different type.

Parameters

__begin1	Begin iterator of first sequence.
__end1	End iterator of first sequence.
__begin2	Begin iterator of second sequence.
__end2	End iterator of second sequence.
__target	Target begin iterator.
__max_length	Maximum number of elements to merge.
__comp	Comparator.

Returns

Output end iterator.

References __merge_advance().

__parallel_nth_element()

```
template<typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_nth_element (
              _RAIter __begin,
              _RAIter __nth,
              _RAIter __end,
              _Compare __comp)
```

Parallel implementation of std::nth_element().

Parameters

__begin	Begin iterator of input sequence.
__nth	_Iterator of element that must be in position afterwards.
__end	End iterator of input sequence.
__comp	Comparator.

References __parallel_partition(), _GLIBCXX_CALL, __gnu_parallel::_Settings::get(), std::max(), __gnu_parallel::_Settings::nth_element_
and __gnu_parallel::_Settings::partition_minimal_n.
Referenced by __parallel_partial_sort().

__parallel_partial_sort()

```
template<typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_partial_sort (
              _RAIter __begin,
              _RAIter __middle,
              _RAIter __end,
              _Compare __comp)
```

Parallel implementation of std::partial_sort().

Parameters

__begin	Begin iterator of input sequence.
__middle	Sort until this position.
__end	End iterator of input sequence.

__comp	Comparator.

References __parallel_nth_element().

__parallel_partial_sum()

```
template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
_OutputIterator __gnu_parallel::__parallel_partial_sum (
            _IIter __begin,
            _IIter __end,
            _OutputIterator __result,
            _BinaryOperation __bin_op)
```

Parallel partial sum front-__end.

Parameters

__begin	Begin iterator of input sequence.
__end	End iterator of input sequence.
__result	Begin iterator of output sequence.
__bin_op	Associative binary function.

Returns

End iterator of output sequence.

References __parallel_partial_sum_linear(), _GLIBCXX_CALL, and __gnu_parallel::_Settings::get().

__parallel_partial_sum_basecase()

```
template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
_OutputIterator __gnu_parallel::__parallel_partial_sum_basecase (
            _IIter __begin,
            _IIter __end,
            _OutputIterator __result,
            _BinaryOperation __bin_op,
            typename std::iterator_traits< _IIter >::value_type __value)
```

Base case prefix sum routine.

Parameters

__begin	Begin iterator of input sequence.
__end	End iterator of input sequence.
__result	Begin iterator of output sequence.
__bin_op	Associative binary function.
__value	Start value. Must be passed since the neutral element is unknown in general.

Returns

End iterator of output sequence.

Referenced by __parallel_partial_sum_linear().

__parallel_partial_sum_linear()

```
template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
_OutputIterator __gnu_parallel::__parallel_partial_sum_linear (
            _IIter __begin,
            _IIter __end,
            _OutputIterator __result,
            _BinaryOperation __bin_op,
            typename std::iterator_traits< _IIter >::difference_type __n)
```

Parallel partial sum implementation, two-phase approach, no recursion.

Parameters

__begin	Begin iterator of input sequence.
__end	End iterator of input sequence.
__result	Begin iterator of output sequence.
__bin_op	Associative binary function.
__n	Length of sequence.

Returns

End iterator of output sequence.

References __equally_split(), __parallel_partial_sum_basecase(), __gnu_parallel::_Settings::get(), std::max(), std::min(), and __gnu_parallel::_Settings::partial_sum_dilation.
Referenced by __parallel_partial_sum().

__parallel_partition()

```
template<typename _RAIter, typename _Predicate>
std::iterator_traits< _RAIter >::difference_type __gnu_parallel::__parallel_partition (
            _RAIter __begin,
            _RAIter __end,
            _Predicate __pred,
            _ThreadIndex __num_threads)
```

Parallel implementation of std::partition.

Parameters

__begin	Begin iterator of input sequence to split.
__end	End iterator of input sequence to split.
__pred	Partition predicate, possibly including some kind of pivot.
__num_threads	Maximum number of threads to use for this task.

Returns

Number of elements not fulfilling the predicate.

References __compare_and_swap(), __fetch_and_add(), _GLIBCXX_CALL, _GLIBCXX_VOLATILE, __gnu_parallel::_Settings::get(), std::max(), __gnu_parallel::_Settings::partition_chunk_share, and __gnu_parallel::_Settings::partition_chunk_size.
Referenced by __parallel_nth_element(), __parallel_sort_qs_divide(), and __qsb_divide().

__parallel_random_shuffle()

```
template<typename _RAIter, typename _RandomNumberGenerator>
void __gnu_parallel::__parallel_random_shuffle (
            _RAIter __begin,
            _RAIter __end,
            _RandomNumberGenerator __rng = _RandomNumber())  [inline]
```

Parallel random public call.

Parameters

__begin	Begin iterator of sequence.
__end	End iterator of sequence.
__rng	Random number generator to use.

References __parallel_random_shuffle_drs().

__parallel_random_shuffle_drs()

```
template<typename _RAIter, typename _RandomNumberGenerator>
void __gnu_parallel::__parallel_random_shuffle_drs (
            _RAIter __begin,
            _RAIter __end,
            typename std::iterator_traits< _RAIter >::difference_type __n,
            _ThreadIndex __num_threads,
            _RandomNumberGenerator & __rng)
```

Main parallel random shuffle step.

Parameters

__begin	Begin iterator of sequence.
__end	End iterator of sequence.
__n	Length of sequence.
__num_threads	Number of threads to use.
__rng	Random number generator to use.

References __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::__bins_end, __parallel_random_shuffle_drs_pu(), __rd_log2(), __round_up_to_pow2(), __sequential_random_shuffle(), _GLIBCXX_CALL, __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_bins_begin, __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_bins_begin, __gnu_parallel::_DRandomShufflingGlobalDa __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_num_bins, __gnu_parallel::_DRandomShufflingGlobalData< _RAIter > __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_num_threads, __gnu_parallel::_DRSSorterPU< _RAIter, __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_seed, __gnu_parallel::_DRandomShufflingGlobalData< _ __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_temporaries, __gnu_parallel::_Settings::get(), __gnu_parallel::_Settings::L2_cache_size, std::max(), std::numeric_limits< _Tp >::max(), std::min(), and __gnu_parallel::_Settings::TLB_

Referenced by __parallel_random_shuffle().

__parallel_random_shuffle_drs_pu()

```
template<typename _RAIter, typename _RandomNumberGenerator>
void __gnu_parallel::__parallel_random_shuffle_drs_pu (
            _DRSSorterPU< _RAIter, _RandomNumberGenerator > * __pus)
```

Random shuffle code executed by each thread.

Parameters

__pus	Array of thread-local data records.

References __random_number_pow2(), __sequential_random_shuffle(), __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M
__gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_bins_begin, __gnu_parallel::_DRandomShufflingGlobalDa
__gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_num_bins, __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >
__gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_num_threads, __gnu_parallel::_DRSSorterPU< _RAIter,
__gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_seed, __gnu_parallel::_DRandomShufflingGlobalData< _
__gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_starts, __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M
and std::partial_sum().
Referenced by __parallel_random_shuffle_drs().

__parallel_sort() [1/7]

```
template<bool __stable, typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_sort (
            _RAIter __begin,
            _RAIter __end,
            _Compare __comp,
            balanced_quicksort_tag __parallelism)  [inline]
```

Choose balanced quicksort for parallel sorting.

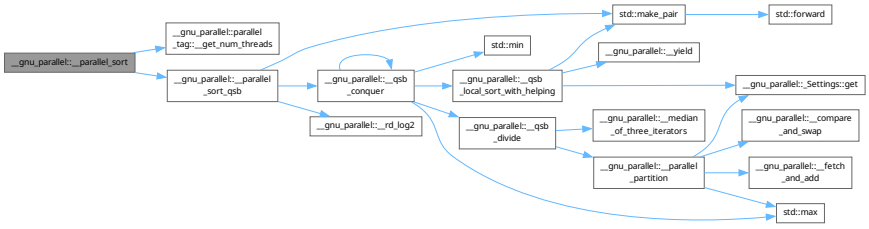
Parameters

__begin	Begin iterator of input sequence.
__end	End iterator of input sequence.
__comp	Comparator.

Template Parameters

__stable	Sort stable.

References __gnu_parallel::parallel_tag::__get_num_threads(), __parallel_sort_qsb(), and _GLIBCXX_CALL.
Here is the call graph for this function:



__parallel_sort() [2/7]

```
template<bool __stable, typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_sort (
```

```
            _RAIter __begin,
            _RAIter __end,
            _Compare __comp,
            default_parallel_tag __parallelism)  [inline]
```

Choose multiway mergesort with exact splitting, for parallel sorting.

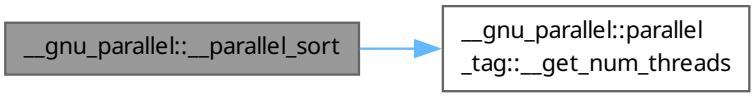
Parameters

__begin	Begin iterator of input sequence.
__end	End iterator of input sequence.
__comp	Comparator.

Template Parameters

__stable	Sort stable.

References __gnu_parallel::parallel_tag::__get_num_threads(), and _GLIBCXX_CALL.
Here is the call graph for this function:



__parallel_sort() [3/7]

```
template<bool __stable, typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_sort (
            _RAIter __begin,
            _RAIter __end,
            _Compare __comp,
            multiway_mergesort_exact_tag __parallelism)  [inline]
```

Choose multiway mergesort with exact splitting, for parallel sorting.

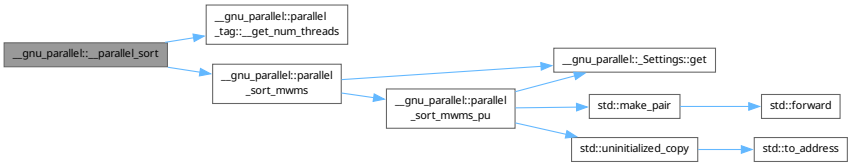
Parameters

__begin	Begin iterator of input sequence.
__end	End iterator of input sequence.
__comp	Comparator.

Template Parameters

__stable	Sort stable.

References __gnu_parallel::parallel_tag::__get_num_threads(), _GLIBCXX_CALL, and parallel_sort_mwms().
Here is the call graph for this function:



__parallel_sort() [4/7]

```
template<bool __stable, typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_sort (
            _RAIter __begin,
            _RAIter __end,
            _Compare __comp,
            multiway_mergesort_sampling_tag __parallelism)  [inline]
```

Choose multiway mergesort with splitting by sampling, for parallel sorting.

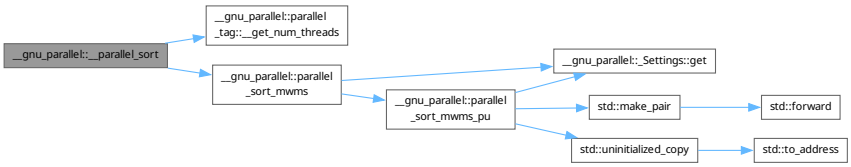
Parameters

__begin	Begin iterator of input sequence.
__end	End iterator of input sequence.
__comp	Comparator.

Template Parameters

__stable	Sort stable.

References __gnu_parallel::parallel_tag::__get_num_threads(), _GLIBCXX_CALL, and parallel_sort_mwms().
Here is the call graph for this function:



__parallel_sort() [5/7]

```
template<bool __stable, typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_sort (
            _RAIter __begin,
            _RAIter __end,
            _Compare __comp,
            multiway_mergesort_tag __parallelism)  [inline]
```

Choose multiway mergesort, splitting variant at run-time, for parallel sorting.

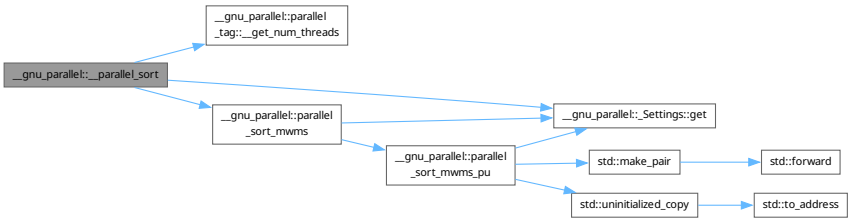
Parameters

__begin	Begin iterator of input sequence.
__end	End iterator of input sequence.
__comp	Comparator.

Template Parameters

__stable	Sort stable.

References __gnu_parallel::parallel_tag::__get_num_threads(), _GLIBCXX_CALL, __gnu_parallel::_Settings::get(), and parallel_sort_mwms().
Here is the call graph for this function:



__parallel_sort() [6/7]

```
template<bool __stable, typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_sort (
            _RAIter __begin,
            _RAIter __end,
            _Compare __comp,
            parallel_tag __parallelism)  [inline]
```

Choose a parallel sorting algorithm.

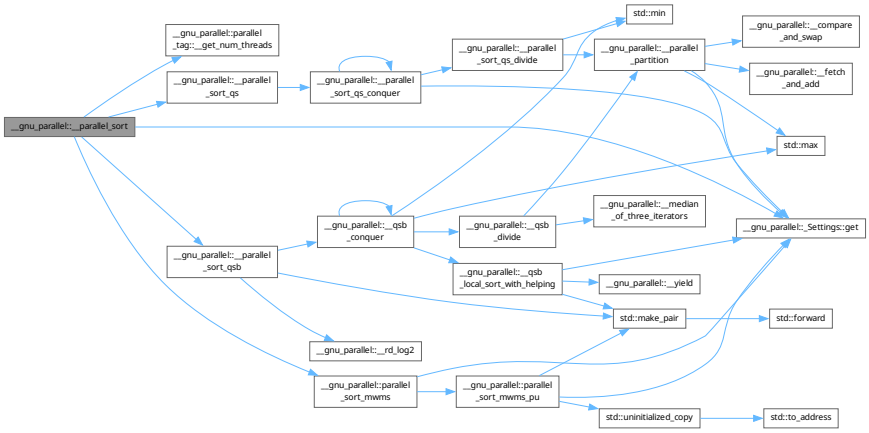
Parameters

__begin	Begin iterator of input sequence.
__end	End iterator of input sequence.
__comp	Comparator.

Template Parameters

__stable	Sort stable.

References __gnu_parallel::parallel_tag::__get_num_threads(), __parallel_sort_qs(), __parallel_sort_qsb(), _GLIBCXX_CALL, __gnu_parallel::_Settings::get(), and parallel_sort_mwms().
Here is the call graph for this function:



__parallel_sort() [7/7]

```
template<bool __stable, typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_sort (
             _RAIter __begin,
             _RAIter __end,
             _Compare __comp,
             quicksort_tag __parallelism)  [inline]
```

Choose quicksort for parallel sorting.

Parameters

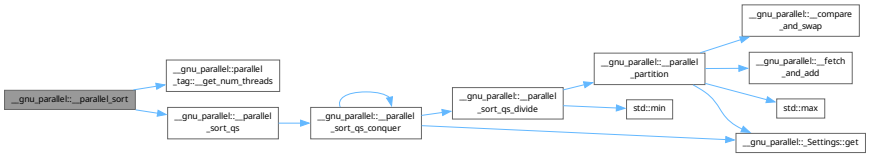
__begin	Begin iterator of input sequence.
__end	End iterator of input sequence.
__comp	Comparator.

Template Parameters

__stable	Sort stable.

References __gnu_parallel::parallel_tag::__get_num_threads(), __parallel_sort_qs(), and _GLIBCXX_CALL.

Here is the call graph for this function:



__parallel_sort_qs()

```
template<typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_sort_qs (
            _RAIter __begin,
            _RAIter __end,
            _Compare __comp,
            _ThreadIndex __num_threads)
```

Unbalanced quicksort main call.

Parameters

__begin	Begin iterator of input sequence.
__end	End iterator input sequence, ignored.
__comp	Comparator.
__num_threads	Number of threads that are allowed to work on this part.

References __parallel_sort_qs_conquer(), and _GLIBCXX_CALL.
Referenced by __parallel_sort(), and __parallel_sort().

__parallel_sort_qs_conquer()

```
template<typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_sort_qs_conquer (
            _RAIter __begin,
            _RAIter __end,
            _Compare __comp,
            _ThreadIndex __num_threads)
```

Unbalanced quicksort conquer step.

Parameters

__begin	Begin iterator of subsequence.
__end	End iterator of subsequence.
__comp	Comparator.
__num_threads	Number of threads that are allowed to work on this part.

References __parallel_sort_qs_conquer(), __parallel_sort_qs_divide(), and __gnu_parallel::_Settings::get().
Referenced by __parallel_sort_qs(), and __parallel_sort_qs_conquer().

__parallel_sort_qs_divide()

```
template<typename _RAIter, typename _Compare>
std::iterator_traits< _RAIter >::difference_type __gnu_parallel::__parallel_sort_qs_divide (
            _RAIter __begin,
            _RAIter __end,
            _Compare __comp,
            typename std::iterator_traits< _RAIter >::difference_type __pivot_rank,
            typename std::iterator_traits< _RAIter >::difference_type __num_samples,
            _ThreadIndex __num_threads)
```

Unbalanced quicksort divide step.

Parameters

__begin	Begin iterator of subsequence.
__end	End iterator of subsequence.
__comp	Comparator.
__pivot_rank	Desired __rank of the pivot.
__num_samples	Choose pivot from that many samples.
__num_threads	Number of threads that are allowed to work on this part.

References __parallel_partition(), and std::min().
Referenced by __parallel_sort_qs_conquer().

__parallel_sort_qsb()

```
template<typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_sort_qsb (
            _RAIter __begin,
            _RAIter __end,
            _Compare __comp,
            _ThreadIndex __num_threads)
```

Top-level quicksort routine.

Parameters

__begin	Begin iterator of sequence.
__end	End iterator of sequence.
__comp	Comparator.
__num_threads	Number of threads that are allowed to work on this part.

References __qsb_conquer(), __rd_log2(), _GLIBCXX_CALL, and std::make_pair().
Referenced by __parallel_sort(), and __parallel_sort().

__parallel_unique_copy() [1/2]

```
template<typename _IIter, class _OutputIterator>
_OutputIterator __gnu_parallel::__parallel_unique_copy (
            _IIter __first,
            _IIter __last,
            _OutputIterator __result)  [inline]
```

Parallel std::unique_copy(), without explicit equality predicate.

Parameters

__first	Begin iterator of input sequence.
__last	End iterator of input sequence.
__result	Begin iterator of result __sequence.

Returns

End iterator of result __sequence.

References __parallel_unique_copy().

__parallel_unique_copy() [2/2]

```
template<typename _IIter, class _OutputIterator, class _BinaryPredicate>
_OutputIterator __gnu_parallel::__parallel_unique_copy (
            _IIter __first,
            _IIter __last,
            _OutputIterator __result,
            _BinaryPredicate __binary_pred)
```

Parallel std::unique_copy(), w/__o explicit equality predicate.

Parameters

__first	Begin iterator of input sequence.
__last	End iterator of input sequence.
__result	Begin iterator of result __sequence.
__binary_pred	Equality predicate.

Returns

End iterator of result __sequence.

References __equally_split(), and _GLIBCXX_CALL.
Referenced by __parallel_unique_copy().

__qsb_conquer()

```
template<typename _RAIter, typename _Compare>
void __gnu_parallel::__qsb_conquer (
            _QSBThreadLocal< _RAIter > ** __tls,
            _RAIter __begin,
            _RAIter __end,
            _Compare __comp,
            _ThreadIndex __iam,
            _ThreadIndex __num_threads,
            bool __parent_wait)
```

Quicksort conquer step.

Parameters

__tls	Array of thread-local storages.
__begin	Begin iterator of subsequence.

Parameters

__end	End iterator of subsequence.
__comp	Comparator.
__iam	Number of the thread processing this function.
__num_threads	Number of threads that are allowed to work on this part.

References __qsb_conquer(), __qsb_divide(), __qsb_local_sort_with_helping(), __gnu_parallel::_QSBThreadLocal< _RAIter >::_M_eler __gnu_parallel::_QSBThreadLocal< _RAIter >::_M_initial, std::max(), and std::min().
Referenced by __parallel_sort_qsb(), and __qsb_conquer().

__qsb_divide()

```
template<typename _RAIter, typename _Compare>
std::iterator_traits< _RAIter >::difference_type __gnu_parallel::__qsb_divide (
            _RAIter __begin,
            _RAIter __end,
            _Compare __comp,
            _ThreadIndex __num_threads)
```

Balanced quicksort divide step.

Parameters

__begin	Begin iterator of subsequence.
__end	End iterator of subsequence.
__comp	Comparator.
__num_threads	Number of threads that are allowed to work on this part.

Precondition

(__end-__begin)>=1

References __median_of_three_iterators(), and __parallel_partition().
Referenced by __qsb_conquer().

__qsb_local_sort_with_helping()

```
template<typename _RAIter, typename _Compare>
void __gnu_parallel::__qsb_local_sort_with_helping (
            _QSBThreadLocal< _RAIter > ** __tls,
            _Compare & __comp,
            _ThreadIndex __iam,
            bool __wait)
```

Quicksort step doing load-balanced local sort.

Parameters

__tls	Array of thread-local storages.
__comp	Comparator.
__iam	Number of the thread processing this function.

References __yield(), _GLIBCXX_PARALLEL_ASSERTIONS, __gnu_parallel::_QSBThreadLocal< _RAIter >::_M_elements_leftover, __gnu_parallel::_QSBThreadLocal< _RAIter >::_M_initial, __gnu_parallel::_QSBThreadLocal< _RAIter >::_M_leftover_parts,

__gnu_parallel::_QSBThreadLocal< _RAIter >::_M_num_threads, __gnu_parallel::_Settings::get(), std::make_pair(), and __gnu_parallel::_Settings::sort_qsb_base_case_maximal_n.
Referenced by __qsb_conquer().

__random_number_pow2()

```
template<typename _RandomNumberGenerator>
int __gnu_parallel::__random_number_pow2 (
            int __logp,
            _RandomNumberGenerator & __rng)  [inline]
```

Generate a random number in [0,2^__logp).

Parameters

__logp	Logarithm (basis 2) of the upper range __bound.
__rng	Random number generator to use.

Referenced by __parallel_random_shuffle_drs_pu(), and __sequential_random_shuffle().

__rd_log2()

```
template<typename _Size>
_Size __gnu_parallel::__rd_log2 (
            _Size __n)  [inline]
```

Calculates the rounded-down logarithm of ___n for base 2.

Parameters

_↩ _n	Argument.

Returns

Returns 0 for any argument <1.

Referenced by __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase(), __parallel_random_shuffle_drs(), __parallel_sort_qsb(), __round_up_to_pow2(), __sequential_random_shuffle(), multiseq_partition(), and multiseq_selection().

__round_up_to_pow2()

```
template<typename _Tp>
_Tp __gnu_parallel::__round_up_to_pow2 (
            _Tp __x)
```

Round up to the next greater power of 2.

Parameters

_↩ _x	_Integer to round up

References __rd_log2().
Referenced by __parallel_random_shuffle_drs(), __sequential_random_shuffle(), and multiseq_selection().

__search_template()

```
template<typename __RAIter1, typename __RAIter2, typename _Pred>
__RAIter1 __gnu_parallel::__search_template (
             __RAIter1 __begin1,
             __RAIter1 __end1,
             __RAIter2 __begin2,
             __RAIter2 __end2,
             _Pred __pred)
```

Parallel std::search.

Parameters

__begin1	Begin iterator of first sequence.
__end1	End iterator of first sequence.
__begin2	Begin iterator of second sequence.
__end2	End iterator of second sequence.
__pred	Find predicate.

Returns

Place of finding in first sequences.

References __calc_borders(), __equally_split(), _GLIBCXX_CALL, std::max(), and std::min().

__sequential_multiway_merge()

```
template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename ↩
_DifferenceTp, typename _Compare>
_RAIter3 __gnu_parallel::__sequential_multiway_merge (
             _RAIterIterator __seqs_begin,
             _RAIterIterator __seqs_end,
             _RAIter3 __target,
             const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator
>::value_type::first_type >::value_type & __sentinel,
             _DifferenceTp __length,
             _Compare __comp)
```

Sequential multi-way merging switch.
The _GLIBCXX_PARALLEL_DECISION is based on the branching factor and runtime settings.

Parameters

__seqs_begin	Begin iterator of iterator pair input sequence.
__seqs_end	End iterator of iterator pair input sequence.
__target	Begin iterator of output sequence.
__comp	Comparator.
__length	Maximum length to merge, possibly larger than the number of elements available.
__sentinel	The sequences have __a __sentinel element.

Returns

End iterator of output sequence.

References __is_sorted(), __merge_advance(), _GLIBCXX_CALL, _GLIBCXX_PARALLEL_LENGTH, and std::min().
Referenced by multiway_merge(), multiway_merge_sentinels(), and parallel_multiway_merge().

__sequential_random_shuffle()

```
template<typename _RAIter, typename _RandomNumberGenerator>
void __gnu_parallel::__sequential_random_shuffle (
             _RAIter __begin,
             _RAIter __end,
             _RandomNumberGenerator & __rng)
```

Sequential cache-efficient random shuffle.

Parameters

__begin	Begin iterator of sequence.
__end	End iterator of sequence.
__rng	Random number generator to use.

References __random_number_pow2(), __rd_log2(), __round_up_to_pow2(), __sequential_random_shuffle(), __gnu_parallel::_Settings::get(), __gnu_parallel::_Settings::L2_cache_size, std::max(), std::min(), std::partial_sum(), and __gnu_parallel::_Settings::TLB_size.
Referenced by __parallel_random_shuffle_drs(), __parallel_random_shuffle_drs_pu(), and __sequential_random_shuffle().

__shrink()

```
template<typename _IIter>
void __gnu_parallel::__shrink (
             std::vector< _IIter > & __os_starts,
             size_t & __count_to_two,
             size_t & __range_length)
```

Combines two ranges into one and thus halves the number of ranges.

Parameters

__os_starts	Start positions worked on (oversampled).
__count_to_two	Counts up to 2.
__range_length	Current length of a chunk.

References std::vector< _Tp, _Alloc >::size().
Referenced by __shrink_and_double().

__shrink_and_double()

```
template<typename _IIter>
void __gnu_parallel::__shrink_and_double (
             std::vector< _IIter > & __os_starts,
             size_t & __count_to_two,
             size_t & __range_length,
             const bool __make_twice)
```

Shrinks and doubles the ranges.

Parameters

__os_starts	Start positions worked on (oversampled).
__count_to_two	Counts up to 2.
__range_length	Current length of a chunk.

__make_twice	Whether the __os_starts is allowed to be grown or not

References __shrink(), std::vector< _Tp, _Alloc >::resize(), and std::vector< _Tp, _Alloc >::size().
Referenced by list_partition().

__yield()

```
void __gnu_parallel::__yield ()  [inline]
```

Yield control to another thread, without waiting for the end of the time slice.
Referenced by __for_each_template_random_access_workstealing(), and __qsb_local_sort_with_helping().

list_partition()

```
template<typename _IIter, typename _FunctorType>
size_t __gnu_parallel::list_partition (
             const _IIter __begin,
             const _IIter __end,
             _IIter * __starts,
             size_t * __lengths,
             const int __num_parts,
             _FunctorType & __f,
             int __oversampling = 0)
```

Splits a sequence given by input iterators into parts of almost equal size.
The function needs only one pass over the sequence.

Parameters

__begin	Begin iterator of input sequence.
__end	End iterator of input sequence.
__starts	Start iterators for the resulting parts, dimension __num_parts+1. For convenience, __starts [__num_parts] contains the end iterator of the sequence.
__lengths	Length of the resulting parts.
__num_parts	Number of parts to split the sequence into.
__f	Functor to be applied to each element by traversing __it
__oversampling	Oversampling factor. If 0, then the partitions will differ in at most $\sqrt{\text{end} - \text{begin}}$ elements. Otherwise, the ratio between the longest and the shortest part is bounded by $1/(\text{oversampling} \cdot \text{num_parts})$

Returns

Length of the whole sequence.

References __shrink_and_double(), and std::vector< _Tp, _Alloc >::size().

max()

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

Equivalent to std::max.
Referenced by __for_each_template_random_access_workstealing().

min()

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

Equivalent to std::min.
Referenced by __for_each_template_random_access_ed(), __for_each_template_random_access_omp_loop(), and __for_each_template_random_access_workstealing().

multiseq_partition()

```
template<typename _RanSeqs, typename _RankType, typename _RankIterator, typename _Compare>
void __gnu_parallel::multiseq_partition (
            _RanSeqs __begin_seqs,
            _RanSeqs __end_seqs,
            _RankType __rank,
            _RankIterator __begin_offsets,
            _Compare __comp = std::less< typename std::iterator_traits<typename std::iterator_traits<↩
_RanSeqs>::value_type:: first_type>::value_type>())
```

Splits several sorted sequences at a certain global __rank, resulting in a splitting point for each sequence. The sequences are passed via a sequence of random-access iterator pairs, none of the sequences may be empty. If there are several equal elements across the split, the ones on the __left side will be chosen from sequences with smaller number.

Parameters

__begin_seqs	Begin of the sequence of iterator pairs.
__end_seqs	End of the sequence of iterator pairs.
__rank	The global rank to partition at.
__begin_offsets	A random-access __sequence __begin where the __result will be stored in. Each element of the sequence is an iterator that points to the first element on the greater part of the respective __sequence.
__comp	The ordering functor, defaults to std::less<_Tp>.

References __rd_log2(), _GLIBCXX_CALL, std::distance(), std::priority_queue< _Tp, _Sequence, _Compare >::empty(), std::make_pair(), std::max(), std::min(), std::priority_queue< _Tp, _Sequence, _Compare >::pop(), std::priority_queue< _Tp, _Sequence and std::priority_queue< _Tp, _Sequence, _Compare >::top().
Referenced by multiway_merge_exact_splitting().

multiseq_selection()

```
template<typename _Tp, typename _RanSeqs, typename _RankType, typename _Compare>
_Tp __gnu_parallel::multiseq_selection (
            _RanSeqs __begin_seqs,
            _RanSeqs __end_seqs,
            _RankType __rank,
            _RankType & __offset,
            _Compare __comp = std::less<_Tp>())
```

Selects the element at a certain global __rank from several sorted sequences.
The sequences are passed via a sequence of random-access iterator pairs, none of the sequences may be empty.

Parameters

__begin_seqs	Begin of the sequence of iterator pairs.

Parameters

__end_seqs	End of the sequence of iterator pairs.
__rank	The global rank to partition at.
__offset	The rank of the selected element in the global subsequence of elements equal to the selected element. If the selected element is unique, this number is 0.
__comp	The ordering functor, defaults to std::less.

References __rd_log2(), __round_up_to_pow2(), _GLIBCXX_CALL, std::distance(), std::priority_queue< _Tp, _Sequence, _Compare >::
std::make_pair(), std::max(), std::min(), std::priority_queue< _Tp, _Sequence, _Compare >::pop(), std::priority_queue< _Tp, _Sequence
and std::priority_queue< _Tp, _Sequence, _Compare >::top().

multiway_merge()

```
template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _↩
Compare>
_RAIterOut __gnu_parallel::multiway_merge (
             _RAIterPairIterator __seqs_begin,
             _RAIterPairIterator __seqs_end,
             _RAIterOut __target,
             _DifferenceTp __length,
             _Compare __comp,
             __gnu_parallel::sequential_tag )
```

Multiway Merge Frontend.
Merge the sequences specified by seqs_begin and __seqs_end into __target. __seqs_begin and __seqs_end must point to a sequence of pairs. These pairs must contain an iterator to the beginning of a sequence in their first entry and an iterator the _M_end of the same sequence in their second entry.
Ties are broken arbitrarily. See stable_multiway_merge for a variant that breaks ties by sequence number but is slower.
The first entries of the pairs (i.e. the begin iterators) will be moved forward.
The output sequence has to provide enough space for all elements that are written to it.
This function will merge the input sequences:

- not stable
- parallel, depending on the input size and Settings
- using sampling for splitting
- not using sentinels

Example:

```
int sequences[10][10];
for (int __i = 0; __i < 10; ++__i)
  for (int __j = 0; __i < 10; ++__j)
    sequences[__i][__j] = __j;

int __out[33];
std::vector<std::pair<int*> > seqs;
for (int __i = 0; __i < 10; ++__i)
  { seqs.push(std::make_pair<int*>(sequences[__i],
                                   sequences[__i] + 10)) }

multiway_merge(seqs.begin(), seqs.end(), __target, std::less<int>(), 33);
```

See also

stable_multiway_merge

Precondition

All input sequences must be sorted.

Target must provide enough space to merge out length elements or the number of elements in all sequences, whichever is smaller.

Postcondition

[__target, return __value) contains merged __elements from the input sequences.

return __value - __target = min(__length, number of elements in all sequences).

Template Parameters

_RAIterPairIterator	iterator over sequence of pairs of iterators
_RAIterOut	iterator over target sequence
_DifferenceTp	difference type for the sequence
_Compare	strict weak ordering type to compare elements in sequences

Parameters

__seqs_begin	__begin of sequence __sequence
__seqs_end	_M_end of sequence __sequence
__target	target sequence to merge to.
__comp	strict weak ordering to use for element comparison.
__length	Maximum length to merge, possibly larger than the number of elements available.

Returns

_M_end iterator of output sequence

References __sequential_multiway_merge(), and _GLIBCXX_CALL.

multiway_merge_3_variant()

```
template<template< typename _RAI, typename _Cp > class iterator, typename _RAIterIterator, typename
_RAIter3, typename _DifferenceTp, typename _Compare>
_RAIter3 __gnu_parallel::multiway_merge_3_variant (
            _RAIterIterator __seqs_begin,
            _RAIterIterator __seqs_end,
            _RAIter3 __target,
            _DifferenceTp __length,
            _Compare __comp)
```

Highly efficient 3-way merging procedure.

Merging is done with the algorithm implementation described by Peter Sanders. Basically, the idea is to minimize the number of necessary comparison after merging an element. The implementation trick that makes this fast is that the order of the sequences is stored in the instruction pointer (translated into labels in C++).

This works well for merging up to 4 sequences.

Note that making the merging stable does *not* come at a performance hit.

Whether the merging is done guarded or unguarded is selected by the used iterator class.

Parameters

__seqs_begin	Begin iterator of iterator pair input sequence.
__seqs_end	End iterator of iterator pair input sequence.
__target	Begin iterator of output sequence.
__comp	Comparator.
__length	Maximum length to merge, less equal than the total number of elements available.

Returns

End iterator of output sequence.

References _GLIBCXX_CALL.

multiway_merge_4_variant()

```
template<template< typename _RAI, typename _Cp > class iterator, typename _RAIterIterator, typename
_RAIter3, typename _DifferenceTp, typename _Compare>
_RAIter3 __gnu_parallel::multiway_merge_4_variant (
            _RAIterIterator __seqs_begin,
            _RAIterIterator __seqs_end,
            _RAIter3 __target,
            _DifferenceTp __length,
            _Compare __comp)
```

Highly efficient 4-way merging procedure.
Merging is done with the algorithm implementation described by Peter Sanders. Basically, the idea is to minimize the number of necessary comparison after merging an element. The implementation trick that makes this fast is that the order of the sequences is stored in the instruction pointer (translated into goto labels in C++).
This works well for merging up to 4 sequences.
Note that making the merging stable does *not* come at a performance hit.
Whether the merging is done guarded or unguarded is selected by the used iterator class.

Parameters

__seqs_begin	Begin iterator of iterator pair input sequence.
__seqs_end	End iterator of iterator pair input sequence.
__target	Begin iterator of output sequence.
__comp	Comparator.
__length	Maximum length to merge, less equal than the total number of elements available.

Returns

End iterator of output sequence.

References _GLIBCXX_CALL.

multiway_merge_exact_splitting()

```
template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType>
void __gnu_parallel::multiway_merge_exact_splitting (
            _RAIterIterator __seqs_begin,
            _RAIterIterator __seqs_end,
            _DifferenceType __length,
```

```
            _DifferenceType __total_length,
            _Compare __comp,
            std::vector< std::pair< _DifferenceType, _DifferenceType > > * __pieces)
```

Exact splitting for parallel multiway-merge routine.
None of the passed sequences may be empty.
References __equally_split(), _GLIBCXX_PARALLEL_LENGTH, std::vector< _Tp, _Alloc >::begin(), std::vector< _Tp, _Alloc >::end(), multiseq_partition(), and std::vector< _Tp, _Alloc >::resize().
Referenced by __parallel_merge_advance().

multiway_merge_loser_tree()

```
template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename
_Compare>
_RAIter3 __gnu_parallel::multiway_merge_loser_tree (
            _RAIterIterator __seqs_begin,
            _RAIterIterator __seqs_end,
            _RAIter3 __target,
            _DifferenceTp __length,
            _Compare __comp)
```

Multi-way merging procedure for a high branching factor, guarded case.
This merging variant uses a LoserTree class as selected by _LT.
Stability is selected through the used LoserTree class _LT.
At least one non-empty sequence is required.

Parameters

__seqs_begin	Begin iterator of iterator pair input sequence.
__seqs_end	End iterator of iterator pair input sequence.
__target	Begin iterator of output sequence.
__comp	Comparator.
__length	Maximum length to merge, less equal than the total number of elements available.

Returns

End iterator of output sequence.

References _GLIBCXX_CALL, and _GLIBCXX_PARALLEL_LENGTH.

multiway_merge_loser_tree_sentinel()

```
template<typename _UnguardedLoserTree, typename _RAIterIterator, typename _RAIter3, typename _↩
DifferenceTp, typename _Compare>
_RAIter3 __gnu_parallel::multiway_merge_loser_tree_sentinel (
            _RAIterIterator __seqs_begin,
            _RAIterIterator __seqs_end,
            _RAIter3 __target,
            const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator
>::value_type::first_type >::value_type & __sentinel,
            _DifferenceTp __length,
            _Compare __comp)
```

Multi-way merging procedure for a high branching factor, requiring sentinels to exist.

Template Parameters

_UnguardedLoserTree	Loser Tree variant to use for the unguarded merging.

Parameters

__seqs_begin	Begin iterator of iterator pair input sequence.
__seqs_end	End iterator of iterator pair input sequence.
__target	Begin iterator of output sequence.
__comp	Comparator.
__length	Maximum length to merge, less equal than the total number of elements available.

Returns

End iterator of output sequence.

References __is_sorted(), _GLIBCXX_CALL, and multiway_merge_loser_tree_unguarded().

multiway_merge_loser_tree_unguarded()

```
template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename
_Compare>
_RAIter3 __gnu_parallel::multiway_merge_loser_tree_unguarded (
            _RAIterIterator __seqs_begin,
            _RAIterIterator __seqs_end,
            _RAIter3 __target,
            const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator
>::value_type::first_type >::value_type & __sentinel,
            _DifferenceTp __length,
            _Compare __comp)
```

Multi-way merging procedure for a high branching factor, unguarded case.
Merging is done using the LoserTree class _LT.
Stability is selected by the used LoserTrees.

Precondition

No input will run out of elements during the merge.

Parameters

__seqs_begin	Begin iterator of iterator pair input sequence.
__seqs_end	End iterator of iterator pair input sequence.
__target	Begin iterator of output sequence.
__comp	Comparator.
__length	Maximum length to merge, less equal than the total number of elements available.

Returns

End iterator of output sequence.

References _GLIBCXX_CALL.
Referenced by multiway_merge_loser_tree_sentinel().

multiway_merge_sampling_splitting()

```
template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType>
void __gnu_parallel::multiway_merge_sampling_splitting (
             _RAIterIterator __seqs_begin,
             _RAIterIterator __seqs_end,
             _DifferenceType __length,
             _DifferenceType __total_length,
             _Compare __comp,
             std::vector< std::pair< _DifferenceType, _DifferenceType > > * __pieces)
```

Sampling based splitting for parallel multiway-merge routine.
References _GLIBCXX_PARALLEL_LENGTH, __gnu_parallel::_Settings::get(), and __gnu_parallel::_Settings::merge_oversampling.

multiway_merge_sentinels()

```
template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _↩
Compare>
_RAIterOut __gnu_parallel::multiway_merge_sentinels (
             _RAIterPairIterator __seqs_begin,
             _RAIterPairIterator __seqs_end,
             _RAIterOut __target,
             _DifferenceTp __length,
             _Compare __comp,
             __gnu_parallel::sequential_tag )
```

Multiway Merge Frontend.
Merge the sequences specified by seqs_begin and __seqs_end into __target. __seqs_begin and __seqs_end must point to a sequence of pairs. These pairs must contain an iterator to the beginning of a sequence in their first entry and an iterator the _M_end of the same sequence in their second entry.
Ties are broken arbitrarily. See stable_multiway_merge for a variant that breaks ties by sequence number but is slower.
The first entries of the pairs (i.e. the begin iterators) will be moved forward accordingly.
The output sequence has to provide enough space for all elements that are written to it.
This function will merge the input sequences:

- not stable
- parallel, depending on the input size and Settings
- using sampling for splitting
- using sentinels

You have to take care that the element the _M_end iterator points to is readable and contains a value that is greater than any other non-sentinel value in all sequences.
Example:

```
  int sequences[10][11];
  for (int __i = 0; __i < 10; ++__i)
    for (int __j = 0; __i < 11; ++__j)
      sequences[__i][__j] = __j; // __last one is sentinel!

  int __out[33];
  std::vector<std::pair<int*> > seqs;
  for (int __i = 0; __i < 10; ++__i)
    { seqs.push(std::make_pair<int*>(sequences[__i],
                                     sequences[__i] + 10)) }

  multiway_merge(seqs.begin(), seqs.end(), __target, std::less<int>(), 33);
```

Precondition

All input sequences must be sorted.

Target must provide enough space to merge out length elements or the number of elements in all sequences, whichever is smaller.

For each __i, __seqs_begin[__i].second must be the end marker of the sequence, but also reference the one more __sentinel element.

Postcondition

[__target, return __value) contains merged __elements from the input sequences.

return __value - __target = min(__length, number of elements in all sequences).

See also

stable_multiway_merge_sentinels

Template Parameters

_RAIterPairIterator	iterator over sequence of pairs of iterators
_RAIterOut	iterator over target sequence
_DifferenceTp	difference type for the sequence
_Compare	strict weak ordering type to compare elements in sequences

Parameters

__seqs_begin	__begin of sequence __sequence
__seqs_end	_M_end of sequence __sequence
__target	target sequence to merge to.
__comp	strict weak ordering to use for element comparison.
__length	Maximum length to merge, possibly larger than the number of elements available.

Returns

_M_end iterator of output sequence

References __sequential_multiway_merge(), and _GLIBCXX_CALL.

parallel_multiway_merge()

```
template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename ↩
_DifferenceTp, typename _Splitter, typename _Compare>
_RAIter3 __gnu_parallel::parallel_multiway_merge (
             _RAIterIterator __seqs_begin,
             _RAIterIterator __seqs_end,
             _RAIter3 __target,
             _Splitter __splitter,
             _DifferenceTp __length,
             _Compare __comp,
             _ThreadIndex __num_threads)
```

Parallel multi-way merge routine.
The _GLIBCXX_PARALLEL_DECISION is based on the branching factor and runtime settings.
Must not be called if the number of sequences is 1.

Template Parameters

_Splitter	functor to split input (either __exact or sampling based)
__stable	Stable merging incurs a performance penalty.
__sentinel	Ignored.

Parameters

__seqs_begin	Begin iterator of iterator pair input sequence.
__seqs_end	End iterator of iterator pair input sequence.
__target	Begin iterator of output sequence.
__comp	Comparator.
__length	Maximum length to merge, possibly larger than the number of elements available.

Returns

End iterator of output sequence.

References __is_sorted(), __sequential_multiway_merge(), _GLIBCXX_CALL, _GLIBCXX_PARALLEL_LENGTH, __gnu_parallel::_Settings::get(), std::make_pair(), __gnu_parallel::_Settings::merge_oversampling, and std::min().
Referenced by __parallel_merge_advance().

parallel_sort_mwms()

```
template<bool __stable, bool __exact, typename _RAIter, typename _Compare>
void __gnu_parallel::parallel_sort_mwms (
            _RAIter __begin,
            _RAIter __end,
            _Compare __comp,
            _ThreadIndex __num_threads)
```

PMWMS main call.

Parameters

__begin	Begin iterator of sequence.
__end	End iterator of sequence.
__comp	Comparator.
__num_threads	Number of threads to use.

References _GLIBCXX_CALL, __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_num_threads, __gnu_parallel::_PMWMSSortingD
__gnu_parallel::_PMWMSSortingData< _RAIter >::_M_pieces, __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_samples,
__gnu_parallel::_PMWMSSortingData< _RAIter >::_M_source, __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_starts,
__gnu_parallel::_PMWMSSortingData< _RAIter >::_M_temporary, __gnu_parallel::_Settings::get(), parallel_sort_mwms_pu(),
and __gnu_parallel::_Settings::sort_mwms_oversampling.
Referenced by __parallel_sort(), __parallel_sort(), __parallel_sort(), and __parallel_sort().

parallel_sort_mwms_pu()

```
template<bool __stable, bool __exact, typename _RAIter, typename _Compare>
void __gnu_parallel::parallel_sort_mwms_pu (
            _PMWMSSortingData< _RAIter > * __sd,
            _Compare & __comp)
```

PMWMS code executed by each thread.

Parameters

__sd	Pointer to algorithm data.
__comp	Comparator.

References __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_num_threads, __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_source, __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_starts, __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_temporary, __gnu_parallel::_Settings::get(), std::make_pair(), __gnu_parallel::_Settings::sort_mwms_oversampling, and std::uninitialized_copy().
Referenced by parallel_sort_mwms().

4.6.5 Variable Documentation

_CASable_bits

```
const int __gnu_parallel::_CASable_bits  [static]
```

Number of bits of _CASable.
Referenced by __decode2(), and __encode2().

_CASable_mask

```
const _CASable __gnu_parallel::_CASable_mask  [static]
```

_CASable with the right half of bits set to 1.
Referenced by __decode2().

4.7 __gnu_pbds Namespace Reference

Classes

- struct associative_tag
- class basic_branch
- struct basic_branch_tag
- class basic_hash_table
- struct basic_hash_tag
- struct basic_invalidation_guarantee
- struct binary_heap_tag
- struct binomial_heap_tag
- class cc_hash_max_collision_check_resize_trigger
- class cc_hash_table
- struct cc_hash_tag
- struct container_error
- struct container_tag
- struct container_traits
- struct container_traits_base
- struct container_traits_base< binary_heap_tag >
- struct container_traits_base< binomial_heap_tag >
- struct container_traits_base< cc_hash_tag >
- struct container_traits_base< gp_hash_tag >
- struct container_traits_base< list_update_tag >
- struct container_traits_base< ov_tree_tag >
- struct container_traits_base< pairing_heap_tag >
- struct container_traits_base< pat_trie_tag >
- struct container_traits_base< rb_tree_tag >

- struct container_traits_base< rc_binomial_heap_tag >
- struct container_traits_base< splay_tree_tag >
- struct container_traits_base< thin_heap_tag >
- class direct_mask_range_hashing
- class direct_mod_range_hashing
- class gp_hash_table
- struct gp_hash_tag
- class hash_exponential_size_policy
- class hash_load_check_resize_trigger
- class hash_prime_size_policy
- class hash_standard_resize_policy
- struct insert_error
- struct join_error
- class linear_probe_fn
- class list_update
- struct list_update_tag
- class lu_counter_policy
- class lu_move_to_front_policy
- struct null_node_update
- struct null_type
- struct ov_tree_tag
- struct pairing_heap_tag
- struct pat_trie_tag
- struct point_invalidation_guarantee
- class priority_queue
- struct priority_queue_tag
- class quadratic_probe_fn
- struct range_invalidation_guarantee
- struct rb_tree_tag
- struct rc_binomial_heap_tag
- struct resize_error
- class sample_probe_fn
- class sample_range_hashing
- class sample_ranged_hash_fn
- class sample_ranged_probe_fn
- class sample_resize_policy
- class sample_resize_trigger
- class sample_size_policy
- class sample_tree_node_update
- struct sample_trie_access_traits
- class sample_trie_node_update
- struct sample_update_policy
- struct sequence_tag
- struct splay_tree_tag
- struct string_tag
- struct thin_heap_tag
- class tree
- class tree_order_statistics_node_update
- struct tree_tag
- class trie
- class trie_order_statistics_node_update

- class trie_prefix_search_node_update
- struct trie_string_access_traits
- struct trie_tag
- struct trivial_iterator_tag

Typedefs

- typedef void trivial_iterator_difference_type

Functions

- void **__throw_container_error** ()
- void **__throw_insert_error** ()
- void **__throw_join_error** ()
- void **__throw_resize_error** ()

Variables

- template<typename String, typename String::value_type Min_E_Val, typename String::value_type Max_E_Val, bool Reverse, typename _Alloc>
  detail::integral_constant< int, Reverse > **trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::s_rev_ind**

4.7.1 Detailed Description

GNU extensions for policy-based data structures for public use.

4.8 __gnu_sequential Namespace Reference

4.8.1 Detailed Description

GNU sequential classes for public use.

4.9 abi Namespace Reference

4.9.1 Detailed Description

The cross-vendor C++ Application Binary Interface. A namespace alias to __cxxabiv1, but user programs should use the alias 'abi'.
A brief overview of an ABI is given in the libstdc++ FAQ, question 5.8 (you may have a copy of the FAQ locally, or you can view the online version at http://gcc.gnu.org/onlinedocs/libstdc++/faq.html#5_8).
GCC subscribes to a cross-vendor ABI for C++, sometimes called the IA64 ABI because it happens to be the native ABI for that platform. It is summarized at http://www.codesourcery.com/cxx-abi/ along with the current specification.
For users of GCC greater than or equal to 3.x, entry points are available in <cxxabi.h>, which notes, *'It is not normally necessary for user programs to include this header, or use the entry points directly. However, this header is available should that be needed.'*

4.10 std Namespace Reference

Namespaces

- namespace __debug
- namespace __detail
- namespace __parallel

- namespace chrono
- namespace decimal
- namespace experimental
- namespace filesystem
- namespace literals
- namespace placeholders
- namespace regex_constants
- namespace rel_ops
- namespace this_thread
- namespace tr1
- namespace tr2

Classes

- class __basic_future
- class __codecvt_abstract_base
- class __ctype_abstract_base
- struct __is_fast_hash
- struct __is_location_invariant
- class __new_allocator
- struct __numeric_limits_base
- struct _Base_bitset
- struct _Base_bitset< 0 >
- struct _Base_bitset< 1 >
- class _Bind
- class _Bind_result
- class _Deque_base
- struct _Deque_iterator
- class _Function_base
- struct _Fwd_list_base
- struct _Fwd_list_const_iterator
- struct _Fwd_list_iterator
- struct _Fwd_list_node
- struct _Fwd_list_node_base
- class _List_base
- struct _List_const_iterator
- struct _List_iterator
- struct _List_node
- class _Not_fn
- struct _Placeholder
- struct _Sp_ebo_helper< _Nm, _Tp, false >
- struct _Sp_ebo_helper< _Nm, _Tp, true >
- class _Temporary_buffer
- struct _Vector_base
- struct add_const
- struct add_cv
- struct add_lvalue_reference
- struct add_pointer
- struct add_rvalue_reference
- struct add_volatile
- struct adopt_lock_t

- struct aligned_storage
- struct aligned_union
- struct alignment_of
- class allocator
- struct allocator_traits
- struct allocator_traits< allocator< _Tp > >
- struct allocator_traits< allocator< void > >
- struct allocator_traits< pmr::polymorphic_allocator< _Tp > >
- struct array
- struct atomic
- struct atomic< _Tp * >
- struct atomic_flag
- class auto_ptr
- struct auto_ptr_ref
- class back_insert_iterator
- class bad_alloc
- class bad_cast
- class bad_exception
- class bad_function_call
- class bad_typeid
- class bad_weak_ptr
- class basic_filebuf
- class basic_fstream
- class basic_ifstream
- class basic_ios
- class basic_iostream
- class basic_istream
- class basic_istringstream
- class basic_ofstream
- class basic_ostream
- class basic_ostringstream
- class basic_regex
- class basic_streambuf
- class basic_string
- class basic_string_view
- class basic_stringbuf
- class basic_stringstream
- class bernoulli_distribution
- struct bidirectional_iterator_tag
- struct binary_function
- class binary_negate
- class binder1st
- class binder2nd
- class binomial_distribution
- class bitset
- class cauchy_distribution
- struct char_traits
- struct char_traits< __gnu_cxx::character< _Value, _Int, _St > >
- struct char_traits< wchar_t >
- class chi_squared_distribution
- class codecvt

- class codecvt< _InternT, _ExternT, encoding_state >
- class codecvt< char, char, mbstate_t >
- class codecvt< char16_t, char, mbstate_t >
- class codecvt< char32_t, char, mbstate_t >
- class codecvt< wchar_t, char, mbstate_t >
- class codecvt_base
- class codecvt_byname
- class common_iterator
- struct common_type
- struct common_type< chrono::duration< _Rep, _Period > >
- struct common_type< chrono::duration< _Rep, _Period >, chrono::duration< _Rep, _Period > >
- struct common_type< chrono::duration< _Rep1, _Period1 >, chrono::duration< _Rep2, _Period2 > >
- struct common_type< chrono::time_point< _Clock, _Duration > >
- struct common_type< chrono::time_point< _Clock, _Duration >, chrono::time_point< _Clock, _Duration > >
- struct common_type< chrono::time_point< _Clock, _Duration1 >, chrono::time_point< _Clock, _Duration2 > >
- struct compare_three_way_result
- class complex
- class complex< double >
- class complex< float >
- class complex< long double >
- class condition_variable
- class condition_variable_any
- struct conditional
- class const_mem_fun1_ref_t
- class const_mem_fun1_t
- class const_mem_fun_ref_t
- class const_mem_fun_t
- struct contiguous_iterator_tag
- class copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>
- class counted_iterator
- class ctype
- class ctype< char >
- class ctype< wchar_t >
- struct ctype_base
- class ctype_byname
- class ctype_byname< char >
- struct decay
- struct default_delete
- struct default_delete< _Tp[ ]>
- struct default_sentinel_t
- struct defer_lock_t
- class deque
- struct destroying_delete_t
- class discard_block_engine
- class discrete_distribution
- struct divides
- struct divides< void >
- class domain_error
- struct enable_if
- class enable_shared_from_this
- class encoding_state

- struct equal_to
- class error_category
- class error_code
- class error_condition
- class exception
- class exception_ptr
- class exponential_distribution
- struct extent
- class extreme_value_distribution
- class fisher_f_distribution
- struct forward_iterator_tag
- class forward_list
- class fpos
- struct from_chars_result
- class front_insert_iterator
- class function< _Res(_ArgTypes...)>
- class function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>
- class future
- class future< _Res & >
- class future< void >
- class future_error
- class gamma_distribution
- class geometric_distribution
- struct greater
- struct greater< void >
- struct greater_equal
- struct greater_equal< void >
- class gslice
- class gslice_array
- struct has_virtual_destructor
- struct hash
- struct hash< __debug::bitset< _Nb > >
- struct hash< __debug::vector< bool, _Alloc > >
- struct hash< __gnu_cxx::__u16vstring >
- struct hash< __gnu_cxx::__u32vstring >
- struct hash< __gnu_cxx::__vstring >
- struct hash< __gnu_cxx::__wvstring >
- struct hash< __gnu_cxx::throw_value_limit >
- struct hash< __gnu_cxx::throw_value_random >
- struct hash< __gnu_debug::basic_string< _CharT > >
- struct hash< __shared_ptr< _Tp, _Lp > >
- struct hash< _Tp * >
- struct hash< basic_string< char, char_traits< char >, _Alloc > >
- struct hash< basic_string< char16_t, char_traits< char16_t >, _Alloc > >
- struct hash< basic_string< char32_t, char_traits< char32_t >, _Alloc > >
- struct hash< basic_string< wchar_t, char_traits< wchar_t >, _Alloc > >
- struct hash< bool >
- struct hash< char >
- struct hash< char16_t >
- struct hash< char32_t >
- struct hash< double >

- struct hash< error_code >
- struct hash< error_condition >
- struct hash< experimental::optional< _Tp > >
- struct hash< experimental::shared_ptr< _Tp > >
- struct hash< float >
- struct hash< int >
- struct hash< long >
- struct hash< long double >
- struct hash< long long >
- struct hash< shared_ptr< _Tp > >
- struct hash< short >
- struct hash< signed char >
- struct hash< thread::id >
- struct hash< type_index >
- struct hash< unique_ptr< _Tp, _Dp > >
- struct hash< unsigned char >
- struct hash< unsigned int >
- struct hash< unsigned long >
- struct hash< unsigned long long >
- struct hash< unsigned short >
- struct hash< wchar_t >
- struct hash<::bitset< _Nb > >
- struct hash<::vector< bool, _Alloc > >
- struct identity
- class independent_bits_engine
- class indirect_array
- class initializer_list
- struct input_iterator_tag
- class insert_iterator
- struct integer_sequence
- struct integral_constant
- class invalid_argument
- class ios_base
- struct is_abstract
- struct is_arithmetic
- struct is_array
- struct is_assignable
- struct is_base_of
- struct is_bind_expression
- struct is_bind_expression< _Bind< _Signature > >
- struct is_bind_expression< _Bind_result< _Result, _Signature > >
- struct is_bind_expression< const _Bind< _Signature > >
- struct is_bind_expression< const _Bind_result< _Result, _Signature > >
- struct is_bind_expression< const volatile _Bind< _Signature > >
- struct is_bind_expression< const volatile _Bind_result< _Result, _Signature > >
- struct is_bind_expression< volatile _Bind< _Signature > >
- struct is_bind_expression< volatile _Bind_result< _Result, _Signature > >
- struct is_class
- struct is_compound
- struct is_const
- struct is_constructible

- struct is_copy_assignable
- struct is_copy_constructible
- struct is_default_constructible
- struct is_destructible
- struct is_empty
- struct is_enum
- struct is_error_code_enum
- struct is_error_code_enum< future_errc >
- struct is_error_condition_enum
- struct is_floating_point
- struct is_function
- struct is_fundamental
- struct is_integral
- struct is_layout_compatible
- struct is_literal_type
- struct is_lvalue_reference
- struct is_member_function_pointer
- struct is_member_object_pointer
- struct is_member_pointer
- struct is_move_assignable
- struct is_move_constructible
- struct is_nothrow_assignable
- struct is_nothrow_constructible
- struct is_nothrow_copy_assignable
- struct is_nothrow_copy_constructible
- struct is_nothrow_default_constructible
- struct is_nothrow_destructible
- struct is_nothrow_move_assignable
- struct is_nothrow_move_constructible
- struct is_object
- struct is_placeholder
- struct is_placeholder< _Placeholder< _Num > >
- struct is_pod
- struct is_pointer
- struct is_pointer_interconvertible_base_of
- struct is_polymorphic
- struct is_reference
- struct is_rvalue_reference
- struct is_same
- struct is_scalar
- struct is_signed
- struct is_standard_layout
- struct is_trivial
- struct is_trivially_assignable
- struct is_trivially_constructible
- struct is_trivially_copy_assignable
- struct is_trivially_copy_constructible
- struct is_trivially_copyable
- struct is_trivially_default_constructible
- struct is_trivially_destructible
- struct is_trivially_move_assignable

- struct is_trivially_move_constructible
- struct is_union
- struct is_unsigned
- struct is_void
- struct is_volatile
- class istream_iterator
- class istreambuf_iterator
- struct iterator
- struct iterator_traits
- struct iterator_traits< _Tp * >
- class length_error
- struct less
- struct less_equal
- struct less_equal< void >
- class linear_congruential_engine
- class list
- class locale
- class lock_guard
- class logic_error
- struct logical_and
- struct logical_and< void >
- struct logical_not
- struct logical_not< void >
- struct logical_or
- struct logical_or< void >
- class lognormal_distribution
- struct make_signed
- struct make_unsigned
- class map
- class mask_array
- class match_results
- class mem_fun1_ref_t
- class mem_fun1_t
- class mem_fun_ref_t
- class mem_fun_t
- class mersenne_twister_engine
- class messages
- struct messages_base
- class messages_byname
- struct minus
- struct minus< void >
- struct modulus
- struct modulus< void >
- class money_base
- class money_get
- class money_put
- class moneypunct
- class moneypunct_byname
- class move_iterator
- class move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>
- class move_sentinel

- class multimap
- struct multiplies
- struct multiplies< void >
- class multiset
- class mutex
- struct negate
- struct negate< void >
- class negative_binomial_distribution
- class nested_exception
- class normal_distribution
- struct not_equal_to
- struct not_equal_to< void >
- class num_get
- class num_put
- struct numeric_limits
- struct numeric_limits< bool >
- struct numeric_limits< char >
- struct numeric_limits< char16_t >
- struct numeric_limits< char32_t >
- struct numeric_limits< double >
- struct numeric_limits< float >
- struct numeric_limits< int >
- struct numeric_limits< long >
- struct numeric_limits< long double >
- struct numeric_limits< long long >
- struct numeric_limits< short >
- struct numeric_limits< signed char >
- struct numeric_limits< unsigned char >
- struct numeric_limits< unsigned int >
- struct numeric_limits< unsigned long >
- struct numeric_limits< unsigned long long >
- struct numeric_limits< unsigned short >
- struct numeric_limits< wchar_t >
- class numpunct
- class numpunct_byname
- struct once_flag
- class ostream_iterator
- class ostreambuf_iterator
- class out_of_range
- struct output_iterator_tag
- class overflow_error
- struct owner_less
- struct owner_less< shared_ptr< _Tp > >
- struct owner_less< void >
- struct owner_less< weak_ptr< _Tp > >
- class packaged_task< _Res(_ArgTypes...)>
- struct pair
- class piecewise_constant_distribution
- struct piecewise_construct_t
- class piecewise_linear_distribution
- struct plus

- class pointer_to_binary_function
- class pointer_to_unary_function
- struct pointer_traits
- struct pointer_traits< _Tp * >
- class poisson_distribution
- class priority_queue
- class promise
- class promise< _Res & >
- class promise< void >
- class queue
- struct random_access_iterator_tag
- class random_device
- class range_error
- struct rank
- struct ratio
- struct ratio_equal
- struct ratio_greater
- struct ratio_greater_equal
- struct ratio_less
- struct ratio_less_equal
- struct ratio_not_equal
- class raw_storage_iterator
- class recursive_mutex
- class recursive_timed_mutex
- class reference_wrapper
- class regex_error
- class regex_iterator
- class regex_token_iterator
- class regex_traits
- struct remove_all_extents
- struct remove_const
- struct remove_cv
- struct remove_extent
- struct remove_pointer
- struct remove_reference
- struct remove_volatile
- struct result_of
- class reverse_iterator
- class runtime_error
- class scoped_allocator_adaptor
- class seed_seq
- class set
- class shared_future
- class shared_future< _Res & >
- class shared_future< void >
- class shared_lock
- class shared_ptr
- class shared_timed_mutex
- class shuffle_order_engine
- class slice
- class slice_array

- class stack
- class student_t_distribution
- class sub_match
- class subtract_with_carry_engine
- class system_error
- class thread
- class time_base
- class time_get
- class time_get_byname
- class time_put
- class time_put_byname
- class timed_mutex
- struct to_chars_result
- struct try_to_lock_t
- class tuple
- class tuple< _T1, _T2 >
- struct tuple_element
- struct tuple_element< 0, pair< _Tp1, _Tp2 > >
- struct tuple_element< 1, pair< _Tp1, _Tp2 > >
- struct tuple_element< __i, tuple< _Types... > >
- struct tuple_element< _Ind, array< _Tp, _Nm > >
- struct tuple_size
- struct tuple_size< array< _Tp, _Nm > >
- struct tuple_size< pair< _Tp1, _Tp2 > >
- struct tuple_size< tuple< _Elements... > >
- struct type_index
- class type_info
- struct unary_function
- class unary_negate
- class underflow_error
- struct underlying_type
- class uniform_int_distribution
- class uniform_real_distribution
- class unique_lock
- class unique_ptr
- class unique_ptr< _Tp[ ], _Dp >
- class unordered_map
- class unordered_multimap
- class unordered_multiset
- class unordered_set
- struct uses_allocator
- struct uses_allocator< tuple< _Types... >, _Alloc >
- class valarray
- class vector
- class vector< bool, _Alloc >
- class wbuffer_convert
- class weak_ptr
- class weibull_distribution
- class wstring_convert

Typedefs

- template<typename _Tp>
  using __allocator_base
- typedef unsigned char **__atomic_flag_data_type**
- typedef FILE **__c_file**
- typedef __locale_t **__c_locale**
- typedef __gthread_mutex_t **__c_lock**
- template<bool _Cond, typename _If, typename _Else>
  using **__conditional_t**
- template<typename _Tp, typename _Up = typename remove_cv<_Tp>::type, typename = typename enable_if<is_same<_Tp, _Up>::value>::type, size_t = tuple_size<_Tp>::value>
  using **__enable_if_has_tuple_size**
- template<bool _Cond, typename _Tp = void>
  using **__enable_if_t**
- template<typename _Fn, typename _Op>
  using **__function_guide_t**
- template<typename _Func, typename _SfinaeType>
  using **__has_is_transparent_t**
- template<typename _ToElementType, typename _FromElementType>
  using **__is_array_convertible**
- template<typename _Tp, typename _Tp2 = typename decay<_Tp>::type>
  using **__is_socketlike**
- template<typename _Iter>
  using **__iter_category_t**
- template<typename _InputIterator>
  using **__iter_key_t**
- template<typename _InputIterator>
  using **__iter_to_alloc_t**
- template<typename _InputIterator>
  using **__iter_val_t**
- template<typename _Tp, typename _Up>
  using **__like_t**
- template<typename _Ptr, typename _Tp>
  using __ptr_rebind
- template<typename _Is, typename _Tp>
  using **__rvalue_stream_extraction_t**
- template<typename _Os, typename _Tp>
  using **__rvalue_stream_insertion_t**
- typedef basic_string< char > **__sso_string**
- template<size_t __i, typename _Tp>
  using **__tuple_element_t**
- template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >, typename _Tr = __umap_traits<__cache_default<_Key, _Hash>::value>>
  using **__umap_hashtable**
- template<bool _Cache>
  using __umap_traits
- template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >, typename _Tr = __ummap_traits<__cache_default<_Key, _Hash>::value>>
  using **__ummap_hashtable**
- template<bool _Cache>
  using __ummap_traits

- template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>, typename _Tr = __umset_traits<__cache_default<_Value, _Hash>::value>>
  using **__umset_hashtable**
- template<bool _Cache>
  using __umset_traits
- template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>, typename _Tr = __uset_traits<__cache_default<_Value, _Hash>::value>>
  using **__uset_hashtable**
- template<bool _Cache>
  using __uset_traits
- template<typename _Fn, typename... _Args>
  using **_Bind_back_t**
- template<typename _Fn, typename... _Args>
  using **_Bind_front_t**
- typedef unsigned long **_Bit_type**
- template<typename _Path, typename _Result = _Path, typename _Path2 = decltype(std::declval<_Path&>().make_preferred().filename())>
  using **_If_fs_path**
- template<typename _InIter>
  using **_RequireInputIter**
- template<typename _Tp>
  using add_lvalue_reference_t
- template<typename _Tp>
  using add_pointer_t
- template<typename _Tp>
  using add_rvalue_reference_t
- template<size_t _Len, size_t _Align = __aligned_storage_default_alignment(_Len)>
  using aligned_storage_t
- template<size_t _Len, typename... _Types>
  using **aligned_union_t**
- typedef atomic< bool > atomic_bool
- typedef atomic< char > atomic_char
- typedef atomic< char16_t > atomic_char16_t
- typedef atomic< char32_t > atomic_char32_t
- typedef atomic< int > atomic_int
- typedef atomic< int16_t > atomic_int16_t
- typedef atomic< int32_t > atomic_int32_t
- typedef atomic< int64_t > atomic_int64_t
- typedef atomic< int8_t > atomic_int8_t
- typedef atomic< int_fast16_t > atomic_int_fast16_t
- typedef atomic< int_fast32_t > atomic_int_fast32_t
- typedef atomic< int_fast64_t > atomic_int_fast64_t
- typedef atomic< int_fast8_t > atomic_int_fast8_t
- typedef atomic< int_least16_t > atomic_int_least16_t
- typedef atomic< int_least32_t > atomic_int_least32_t
- typedef atomic< int_least64_t > atomic_int_least64_t
- typedef atomic< int_least8_t > atomic_int_least8_t
- typedef atomic< intmax_t > atomic_intmax_t
- typedef atomic< intptr_t > atomic_intptr_t
- typedef atomic< long long > atomic_llong
- typedef atomic< long > atomic_long
- typedef atomic< ptrdiff_t > atomic_ptrdiff_t
- typedef atomic< signed char > atomic_schar

- typedef atomic< short > atomic_short
- typedef atomic< size_t > atomic_size_t
- typedef atomic< unsigned char > atomic_uchar
- typedef atomic< unsigned int > atomic_uint
- typedef atomic< uint16_t > atomic_uint16_t
- typedef atomic< uint32_t > atomic_uint32_t
- typedef atomic< uint64_t > atomic_uint64_t
- typedef atomic< uint8_t > atomic_uint8_t
- typedef atomic< uint_fast16_t > atomic_uint_fast16_t
- typedef atomic< uint_fast32_t > atomic_uint_fast32_t
- typedef atomic< uint_fast64_t > atomic_uint_fast64_t
- typedef atomic< uint_fast8_t > atomic_uint_fast8_t
- typedef atomic< uint_least16_t > atomic_uint_least16_t
- typedef atomic< uint_least32_t > atomic_uint_least32_t
- typedef atomic< uint_least64_t > atomic_uint_least64_t
- typedef atomic< uint_least8_t > atomic_uint_least8_t
- typedef atomic< uintmax_t > atomic_uintmax_t
- typedef atomic< uintptr_t > atomic_uintptr_t
- typedef atomic< unsigned long long > atomic_ullong
- typedef atomic< unsigned long > atomic_ulong
- typedef atomic< unsigned short > atomic_ushort
- typedef atomic< wchar_t > atomic_wchar_t
- using **atto**
- using **centi**
- typedef match_results< const char * > **cmatch**
- template<typename... _Ts>
  using **common_comparison_category_t**
- template<typename... _Tp>
  using common_type_t
- template<typename _Tp, typename _Up = _Tp>
  using compare_three_way_result_t
- template<bool _Cond, typename _Iftrue, typename _Iffalse>
  using conditional_t
- typedef regex_iterator< const char * > **cregex_iterator**
- typedef regex_token_iterator< const char * > cregex_token_iterator
- typedef sub_match< const char * > csub_match
- using **deca**
- template<typename _Tp>
  using decay_t
- using **deci**
- typedef minstd_rand0 **default_random_engine**
- template<bool _Cond, typename _Tp = void>
  using enable_if_t
- using **exa**
- using false_type
- using **femto**
- typedef basic_filebuf< char > filebuf
- typedef basic_fstream< char > fstream
- using **giga**
- using **hecto**
- typedef basic_ifstream< char > ifstream

- template<size_t... _Idx>
  using index_sequence
- template<typename... _Types>
  using index_sequence_for
- typedef basic_ios< char > ios
- typedef basic_iostream< char > iostream
- typedef basic_istream< char > istream
- typedef basic_istringstream< char > istringstream
- using **kilo**
- typedef shuffle_order_engine< minstd_rand0, 256 > **knuth_b**
- template<size_t _Num>
  using make_index_sequence
- template<typename _Tp, _Tp _Num>
  using make_integer_sequence
- template<typename _Tp>
  using make_signed_t
- template<typename _Tp>
  using make_unsigned_t
- using **mega**
- using **micro**
- using **milli**
- typedef linear_congruential_engine< uint_fast32_t, 48271UL, 0UL, 2147483647UL > minstd_rand
- typedef linear_congruential_engine< uint_fast32_t, 16807UL, 0UL, 2147483647UL > minstd_rand0
- typedef mersenne_twister_engine< uint_fast32_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL > mt19937
- typedef mersenne_twister_engine< uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67fffeda60000ULL, 37, 0xfff7eee000000000ULL, 43, 6364136223846793005ULL > mt19937_64
- using **nano**
- typedef void(* new_handler) ()
- typedef decltype(nullptr) **nullptr_t**
- typedef basic_ofstream< char > ofstream
- typedef basic_ostream< char > ostream
- typedef basic_ostringstream< char > ostringstream
- using **peta**
- using **pico**
- typedef __PTRDIFF_TYPE__ **ptrdiff_t**
- typedef discard_block_engine< ranlux24_base, 223, 23 > **ranlux24**
- typedef subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 > **ranlux24_base**
- typedef discard_block_engine< ranlux48_base, 389, 11 > **ranlux48**
- typedef subtract_with_carry_engine< uint_fast64_t, 48, 5, 12 > **ranlux48_base**
- template<typename _R1, typename _R2>
  using ratio_add
- template<typename _R1, typename _R2>
  using ratio_divide
- template<typename _R1, typename _R2>
  using ratio_multiply
- template<typename _R1, typename _R2>
  using ratio_subtract
- typedef basic_regex< char > regex
- template<typename _Tp>
  using remove_all_extents_t

- template<typename _Tp>
  using remove_extent_t
- template<typename _Tp>
  using remove_pointer_t
- template<typename _Tp>
  using remove_reference_t
- template<typename _Tp>
  using result_of_t
- typedef __SIZE_TYPE__ **size_t**
- typedef match_results< string::const_iterator > **smatch**
- typedef regex_iterator< string::const_iterator > **sregex_iterator**
- typedef regex_token_iterator< string::const_iterator > sregex_token_iterator
- typedef sub_match< string::const_iterator > ssub_match
- typedef basic_streambuf< char > streambuf
- typedef long long streamoff
- typedef fpos< mbstate_t > streampos
- typedef ptrdiff_t streamsize
- typedef basic_string< char > string
- using **string_view**
- typedef basic_stringbuf< char > stringbuf
- typedef basic_stringstream< char > stringstream
- using **tera**
- typedef void(* terminate_handler) ()
- using true_type
- template<size_t __i, typename _Tp>
  using **tuple_element_t**
- typedef fpos< mbstate_t > u16streampos
- typedef basic_string< char16_t > u16string
- using **u16string_view**
- typedef fpos< mbstate_t > u32streampos
- typedef basic_string< char32_t > u32string
- using **u32string_view**
- template<typename _Tp>
  using underlying_type_t
- typedef void(* unexpected_handler) ()
- typedef match_results< const wchar_t * > **wcmatch**
- typedef regex_iterator< const wchar_t * > **wcregex_iterator**
- typedef regex_token_iterator< const wchar_t * > wcregex_token_iterator
- typedef sub_match< const wchar_t * > wcsub_match
- typedef basic_filebuf< wchar_t > wfilebuf
- typedef basic_fstream< wchar_t > wfstream
- typedef basic_ifstream< wchar_t > wifstream
- typedef basic_ios< wchar_t > wios
- typedef basic_iostream< wchar_t > wiostream
- typedef basic_istream< wchar_t > wistream
- typedef basic_istringstream< wchar_t > wistringstream
- typedef basic_ofstream< wchar_t > wofstream
- typedef basic_ostream< wchar_t > wostream
- typedef basic_ostringstream< wchar_t > wostringstream
- typedef basic_regex< wchar_t > wregex
- typedef match_results< wstring::const_iterator > **wsmatch**

- typedef regex_iterator< wstring::const_iterator > **wsregex_iterator**
- typedef regex_token_iterator< wstring::const_iterator > wsregex_token_iterator
- typedef sub_match< wstring::const_iterator > wssub_match
- typedef basic_streambuf< wchar_t > wstreambuf
- typedef fpos< mbstate_t > wstreampos
- typedef basic_string< wchar_t > wstring
- using **wstring_view**
- typedef basic_stringbuf< wchar_t > wstringbuf
- typedef basic_stringstream< wchar_t > wstringstream

Enumerations

- enum { **_S_chunk_size** }
- enum { **_S_word_bit** }
- enum **_Ios_Fmtflags** {
  _S_boolalpha , **_S_dec** , **_S_fixed** , **_S_hex** ,
  _S_internal , **_S_left** , **_S_oct** , **_S_right** ,
  _S_scientific , **_S_showbase** , **_S_showpoint** , **_S_showpos** ,
  _S_skipws , **_S_unitbuf** , **_S_uppercase** , **_S_adjustfield** ,
  _S_basefield , **_S_floatfield** , **_S_ios_fmtflags_end** , **_S_ios_fmtflags_max** ,
  _S_ios_fmtflags_min }
- enum **_Ios_Iostate** {
  _S_goodbit , **_S_badbit** , **_S_eofbit** , **_S_failbit** ,
  _S_ios_iostate_end , **_S_ios_iostate_max** , **_S_ios_iostate_min** }
- enum **_Ios_Openmode** {
  _S_app , **_S_ate** , **_S_bin** , **_S_in** ,
  _S_out , **_S_trunc** , **_S_noreplace** , **_S_ios_openmode_end** ,
  _S_ios_openmode_max , **_S_ios_openmode_min** }
- enum **_Ios_Seekdir** { **_S_beg** , **_S_cur** , **_S_end** , **_S_ios_seekdir_end** }
- enum **_Lock_policy**
- enum **_Manager_operation** { **__get_type_info** , **__get_functor_ptr** , **__clone_functor** , **__destroy_functor** }
- enum **_Rb_tree_color** { **_S_red** , **_S_black** }
- enum class **align_val_t** : size_t
- enum class chars_format { **scientific** , **fixed** , **hex** , **general** }
- enum **codecvt_mode** { **consume_header** , **generate_header** , **little_endian** }
- enum class cv_status { **no_timeout** , **timeout** }
- enum class **errc** {
  address_family_not_supported , **address_in_use** , **address_not_available** , **already_connected** ,
  argument_list_too_long , **argument_out_of_domain** , **bad_address** , **bad_file_descriptor** ,
  broken_pipe , **connection_aborted** , **connection_already_in_progress** , **connection_refused** ,
  connection_reset , **cross_device_link** , **destination_address_required** , **device_or_resource_busy** ,
  directory_not_empty , **executable_format_error** , **file_exists** , **file_too_large** ,
  filename_too_long , **function_not_supported** , **host_unreachable** , **illegal_byte_sequence** ,
  inappropriate_io_control_operation , **interrupted** , **invalid_argument** , **invalid_seek** ,
  io_error , **is_a_directory** , **message_size** , **network_down** ,
  network_reset , **network_unreachable** , **no_buffer_space** , **no_child_process** ,
  no_lock_available , **no_message** , **no_protocol_option** , **no_space_on_device** ,
  no_such_device_or_address , **no_such_device** , **no_such_file_or_directory** , **no_such_process** ,
  not_a_directory , **not_a_socket** , **not_connected** , **not_enough_memory** ,
  operation_in_progress , **operation_not_permitted** , **operation_not_supported** , **operation_would_block** ,
  permission_denied , **protocol_not_supported** , **read_only_file_system** , **resource_deadlock_would_occur**
  ,
  resource_unavailable_try_again , **result_out_of_range** , **timed_out** , **too_many_files_open_in_system** ,
  too_many_files_open , **too_many_links** , **too_many_symbolic_link_levels** , **wrong_protocol_type** }

- enum float_denorm_style { denorm_indeterminate , denorm_absent , denorm_present }
- enum float_round_style {
  round_indeterminate , round_toward_zero , round_to_nearest , round_toward_infinity ,
  round_toward_neg_infinity }
- enum class future_errc { **future_already_retrieved** , **promise_already_satisfied** , **no_state** , **broken_promise** }
- enum class future_status { **ready** , **timeout** , **deferred** }
- enum class io_errc { **stream** }
- enum class launch { **async** , **deferred** }
- enum class memory_order : int {
  relaxed , **consume** , **acquire** , **release** ,
  acq_rel , **seq_cst** }
- enum class pointer_safety { **relaxed** , **preferred** , **strict** }

Functions

- template<typename _CharT>
  _CharT * **__add_grouping** (_CharT *__s, _CharT __sep, const char *__gbeg, size_t __gsize, const _CharT *__first, const _CharT *__last)
- template<typename _Tp>
  constexpr _Tp * __addressof (_Tp &__r) noexcept
- template<typename _ForwardIterator, typename _BinaryPredicate>
  constexpr _ForwardIterator **__adjacent_find** (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)
- template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _Compare>
  constexpr void **__adjust_heap** (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __len, _Tp __value, _Compare __comp)
- template<typename _BidirectionalIterator, typename _Distance>
  constexpr void **__advance** (_BidirectionalIterator &__i, _Distance __n, bidirectional_iterator_tag)
- template<typename _InputIterator, typename _Distance>
  constexpr void **__advance** (_InputIterator &__i, _Distance __n, input_iterator_tag)
- template<typename _OutputIterator, typename _Distance>
  void **__advance** (_OutputIterator &, _Distance, output_iterator_tag)=delete
- template<typename _RandomAccessIterator, typename _Distance>
  constexpr void **__advance** (_RandomAccessIterator &__i, _Distance __n, random_access_iterator_tag)
- template<typename _Tp, _Lock_policy _Lp = __default_lock_policy, typename _Alloc, typename... _Args>
  __shared_ptr< _Tp, _Lp > **__allocate_shared** (const _Alloc &__a, _Args &&... __args)
- template<bool _IsMove, typename _OutIter, typename _InIter>
  constexpr void **__assign_one** (_OutIter &__out, _InIter &__in)
- template<typename _Facet>
  const _Facet & **__check_facet** (const _Facet *__f)
- template<typename _RandomAccessIterator, typename _Distance, typename _Compare>
  constexpr void **__chunk_insertion_sort** (_RandomAccessIterator __first, _RandomAccessIterator __last, _↩
  Distance __chunk_size, _Compare __comp)
- template<typename _Tp>
  _Tp **__complex_abs** (const complex< _Tp > &__z)
- __complex__ double **__complex_acos** (__complex__ double __z)
- __complex__ float **__complex_acos** (__complex__ float __z)
- __complex__ long double **__complex_acos** (const __complex__ long double &__z)
- template<typename _Tp>
  std::complex< _Tp > **__complex_acos** (const std::complex< _Tp > &__z)
- __complex__ double **__complex_acosh** (__complex__ double __z)

- __complex__ float **__complex_acosh** (__complex__ float __z)
- __complex__ long double **__complex_acosh** (const __complex__ long double &__z)
- template<typename _Tp>
  std::complex< _Tp > **__complex_acosh** (const std::complex< _Tp > &__z)
- template<typename _Tp>
  _Tp **__complex_arg** (const complex< _Tp > &__z)
- __complex__ double **__complex_asin** (__complex__ double __z)
- __complex__ float **__complex_asin** (__complex__ float __z)
- __complex__ long double **__complex_asin** (const __complex__ long double &__z)
- template<typename _Tp>
  std::complex< _Tp > **__complex_asin** (const std::complex< _Tp > &__z)
- __complex__ double **__complex_asinh** (__complex__ double __z)
- __complex__ float **__complex_asinh** (__complex__ float __z)
- __complex__ long double **__complex_asinh** (const __complex__ long double &__z)
- template<typename _Tp>
  std::complex< _Tp > **__complex_asinh** (const std::complex< _Tp > &__z)
- __complex__ double **__complex_atan** (__complex__ double __z)
- __complex__ float **__complex_atan** (__complex__ float __z)
- __complex__ long double **__complex_atan** (const __complex__ long double &__z)
- template<typename _Tp>
  std::complex< _Tp > **__complex_atan** (const std::complex< _Tp > &__z)
- __complex__ double **__complex_atanh** (__complex__ double __z)
- __complex__ float **__complex_atanh** (__complex__ float __z)
- __complex__ long double **__complex_atanh** (const __complex__ long double &__z)
- template<typename _Tp>
  std::complex< _Tp > **__complex_atanh** (const std::complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **__complex_cos** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **__complex_cosh** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **__complex_exp** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **__complex_log** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **__complex_pow** (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp>
  complex< _Tp > **__complex_pow_unsigned** (complex< _Tp > __x, unsigned __n)
- complex< double > **__complex_proj** (const complex< double > &__z)
- complex< float > **__complex_proj** (const complex< float > &__z)
- complex< long double > **__complex_proj** (const complex< long double > &__z)
- template<typename _Tp>
  std::complex< _Tp > **__complex_proj** (const std::complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **__complex_sin** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **__complex_sinh** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **__complex_sqrt** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **__complex_tan** (const complex< _Tp > &__z)

- template<typename _Tp>
  complex< _Tp > **__complex_tanh** (const complex< _Tp > &__z)
- int **__convert_from_v** (const __c_locale &__cloc, char *__out, const int __size, const char *__fmt,...)
- template<typename _Tp>
  void **__convert_to_v** (const char *, _Tp &, ios_base::iostate &, const __c_locale &) throw ()
- template<> void **__convert_to_v** (const char *, double &, ios_base::iostate &, const __c_locale &) throw ()
- template<> void **__convert_to_v** (const char *, float &, ios_base::iostate &, const __c_locale &) throw ()
- template<> void **__convert_to_v** (const char *, long double &, ios_base::iostate &, const __c_locale &) throw ()
- template<bool _IsMove, typename _II, typename _OI>
  constexpr _OI **__copy_move_a** (_II __first, _II __last, _OI __result)
- template<bool _IsMove, typename _II, typename _Ite, typename _Seq, typename _Cat>
  constexpr __gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > **__copy_move_a** (_II, _II, const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &)
- template<bool _IsMove, typename _IIte, typename _ISeq, typename _ICat, typename _OIte, typename _OSeq, typename _OCat>
  constexpr ::__gnu_debug::_Safe_iterator< _OIte, _OSeq, _OCat > **__copy_move_a** (const ::__gnu_debug::_Safe_iterator< _IIte, _ISeq, _ICat > &, const ::__gnu_debug::_Safe_iterator< _IIte, _ISeq, _ICat > &, const ::__gnu_debug::_Safe_iterator< _OIte, _OSeq, _OCat > &)
- template<bool _IsMove, typename _Ite, typename _Seq, typename _Cat, typename _OI>
  constexpr _OI **__copy_move_a** (const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &, const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &, _OI)
- template<bool _IsMove, typename _ITp, typename _IRef, typename _IPtr, typename _OTp>
  ::_Deque_iterator< _OTp, _OTp &, _OTp * > **__copy_move_a1** (::_Deque_iterator< _ITp, _IRef, _IPtr > __first, ::_Deque_iterator< _ITp, _IRef, _IPtr > __last, ::_Deque_iterator< _OTp, _OTp &, _OTp * > __result)
- template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI>
  _OI **__copy_move_a1** (::_Deque_iterator< _Tp, _Ref, _Ptr > __first, ::_Deque_iterator< _Tp, _Ref, _Ptr > __last, _OI __result)
- template<bool _IsMove, typename _II, typename _Tp>
  __gnu_cxx::__enable_if< __is_random_access_iter< _II >::__value,::_Deque_iterator< _Tp, _Tp &, _Tp * > >::__type **__copy_move_a1** (_II __first, _II __last, ::_Deque_iterator< _Tp, _Tp &, _Tp * > __result)
- template<bool _IsMove, typename _II, typename _OI>
  constexpr _OI **__copy_move_a1** (_II __first, _II __last, _OI __result)
- template<bool _IsMove, typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT, char_traits< _CharT > > >::__type **__copy_move_a2** (_CharT *, _CharT *, ostreambuf_iterator< _CharT, char_traits< _CharT > >)
- template<bool _IsMove, typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type **__copy_move_a2** (_CharT *__first, _CharT *__last, ostreambuf_iterator< _CharT > __result)
- template<bool _IsMove, typename _InIter, typename _Sent, typename _OutIter>
  constexpr _OutIter **__copy_move_a2** (_InIter __first, _Sent __last, _OutIter __result)
- template<bool _IsMove, typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT, char_traits< _CharT > > >::__type **__copy_move_a2** (const _CharT *, const _CharT *, ostreambuf_iterator< _CharT, char_traits< _CharT > >)
- template<bool _IsMove, typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type **__copy_move_a2** (const _CharT *__first, const _CharT *__last, ostreambuf_iterator< _CharT > __result)
- template<bool _IsMove, typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type **__copy_move_a2** (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, _CharT *__result)
- template<bool _IsMove, typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value,::_Deque_iterator< _CharT, _CharT &, _CharT * > >::__type **__copy_move_a2** (istreambuf_iterator< _CharT, char_traits< _CharT > > __first,

istreambuf_iterator< _CharT, char_traits< _CharT > > __last, ::_Deque_iterator< _CharT, _CharT &, _CharT * > __result)

- template<bool _IsMove, typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type **__copy_move_a2** (istreambuf_iterator< _CharT, char_traits< _CharT > >, istreambuf_iterator< _CharT, char_traits< _CharT > >, _CharT *)
- template<bool _IsMove, typename _II, typename _OI>
  constexpr _OI **__copy_move_backward_a** (_II __first, _II __last, _OI __result)
- template<bool _IsMove, typename _II, typename _Ite, typename _Seq, typename _Cat>
  constexpr __gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > **__copy_move_backward_a** (_II, _II, const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &)
- template<bool _IsMove, typename _IIte, typename _ISeq, typename _ICat, typename _OIte, typename _OSeq, typename _OCat>
  constexpr ::__gnu_debug::_Safe_iterator< _OIte, _OSeq, _OCat > **__copy_move_backward_a** (const ::__gnu_debug::_Safe_iterator< _IIte, _ISeq, _ICat > &, const ::__gnu_debug::_Safe_iterator< _IIte, _ISeq, _ICat > &, const ::__gnu_debug::_Safe_iterator< _OIte, _OSeq, _OCat > &)
- template<bool _IsMove, typename _Ite, typename _Seq, typename _Cat, typename _OI>
  constexpr _OI **__copy_move_backward_a** (const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &, const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &, _OI)
- template<bool _IsMove, typename _ITp, typename _IRef, typename _IPtr, typename _OTp>
  ::_Deque_iterator< _OTp, _OTp &, _OTp * > **__copy_move_backward_a1** (::_Deque_iterator< _ITp, _IRef, _IPtr > __first, ::_Deque_iterator< _ITp, _IRef, _IPtr > __last, ::_Deque_iterator< _OTp, _OTp &, _OTp * > __result)
- template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI>
  _OI **__copy_move_backward_a1** (::_Deque_iterator< _Tp, _Ref, _Ptr > __first, ::_Deque_iterator< _Tp, _Ref, _Ptr > __last, _OI __result)
- template<bool _IsMove, typename _BI1, typename _BI2>
  constexpr _BI2 **__copy_move_backward_a1** (_BI1 __first, _BI1 __last, _BI2 __result)
- template<bool _IsMove, typename _II, typename _Tp>
  __gnu_cxx::__enable_if< __is_random_access_iter< _II >::__value,::_Deque_iterator< _Tp, _Tp &, _Tp * > >::__type **__copy_move_backward_a1** (_II __first, _II __last, ::_Deque_iterator< _Tp, _Tp &, _Tp * > __result)
- template<bool _IsMove, typename _BI1, typename _BI2>
  constexpr _BI2 **__copy_move_backward_a2** (_BI1 __first, _BI1 __last, _BI2 __result)
- template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI>
  _OI **__copy_move_backward_dit** (::_Deque_iterator< _Tp, _Ref, _Ptr > __first, ::_Deque_iterator< _Tp, _Ref, _Ptr > __last, _OI __result)
- template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI>
  _OI **__copy_move_dit** (::_Deque_iterator< _Tp, _Ref, _Ptr > __first, ::_Deque_iterator< _Tp, _Ref, _Ptr > __last, _OI __result)
- template<typename _InputIterator, typename _Size, typename _OutputIterator>
  constexpr _OutputIterator **__copy_n_a** (_InputIterator __first, _Size __n, _OutputIterator __result, bool)
- template<typename _CharT, typename _Size>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type **__copy_n_a** (istreambuf_iterator< _CharT > __it, _Size __n, _CharT *__result, bool __strict)
- template<typename _CharT, typename _Size>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value,::_Deque_iterator< _CharT, _CharT &, _CharT * > >::__type **__copy_n_a** (istreambuf_iterator< _CharT, char_traits< _CharT > > __it, _Size __size, ::_Deque_iterator< _CharT, _CharT &, _CharT * > __result, bool __strict)
- template<typename _CharT, typename _Size>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type **__copy_n_a** (istreambuf_iterator< _CharT, char_traits< _CharT > >, _Size, _CharT *, bool)
- template<typename _CharT, typename _Traits>
  streamsize **__copy_streambufs** (basic_streambuf< _CharT, _Traits > *__sbin, basic_streambuf< _CharT, _Traits > *__sbout)

- template<typename _CharT, typename _Traits>
  streamsize **__copy_streambufs_eof** (basic_streambuf< _CharT, _Traits > *__sbin, basic_streambuf< _CharT, _Traits > *__sbout, bool &__ineof)
- template<> streamsize **__copy_streambufs_eof** (basic_streambuf< char > *__sbin, basic_streambuf< char > *__sbout, bool &__ineof)
- template<> streamsize **__copy_streambufs_eof** (basic_streambuf< wchar_t > *__sbin, basic_streambuf< wchar_t > *__sbout, bool &__ineof)
- template<typename _InputIterator, typename _Predicate>
  constexpr iterator_traits< _InputIterator >::difference_type **__count_if** (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- constexpr size_t **__deque_buf_size** (size_t __size)
- template<typename _Tp>
  ptrdiff_t **__distance** (::_List_const_iterator< _Tp >, ::_List_const_iterator< _Tp >, input_iterator_tag)
- template<typename _Tp>
  ptrdiff_t **__distance** (::_List_iterator< _Tp >, ::_List_iterator< _Tp >, input_iterator_tag)
- template<bool _Const, typename _Ptr>
  ptrdiff_t **__distance** (__list::_Iterator< _Const, _Ptr > __first, __list::_Iterator< _Const, _Ptr > __last, input_iterator_tag __tag)
- template<typename _InputIterator>
  constexpr iterator_traits< _InputIterator >::difference_type **__distance** (_InputIterator __first, _InputIterator __last, input_iterator_tag)
- template<typename _OutputIterator>
  void **__distance** (_OutputIterator, _OutputIterator, output_iterator_tag)=delete
- template<typename _RandomAccessIterator>
  constexpr iterator_traits< _RandomAccessIterator >::difference_type **__distance** (_RandomAccessIterator __first, _RandomAccessIterator __last, random_access_iterator_tag)
- template<typename _OutStr, typename _InChar, typename _Codecvt, typename _State, typename _Fn>
  bool **__do_str_codecvt** (const _InChar *__first, const _InChar *__last, _OutStr &__outstr, const _Codecvt &__cvt, _State &__state, size_t &__count, _Fn __fn)
- template<typename _II1, typename _II2>
  constexpr bool **__equal4** (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)
- template<typename _II1, typename _II2, typename _BinaryPredicate>
  constexpr bool **__equal4** (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, _BinaryPredicate __binary_pred)
- template<typename _II1, typename _II2>
  constexpr bool **__equal_aux** (_II1 __first1, _II1 __last1, _II2 __first2)
- template<typename _II1, typename _II2, typename _Seq2, typename _Cat2>
  constexpr bool **__equal_aux** (_II1, _II1, const ::__gnu_debug::_Safe_iterator< _II2, _Seq2, _Cat2 > &)
- template<typename _II1, typename _Seq1, typename _Cat1, typename _II2>
  constexpr bool **__equal_aux** (const ::__gnu_debug::_Safe_iterator< _II1, _Seq1, _Cat1 > &, const ::__gnu_debug::_Safe_iterator< _II1, _Seq1, _Cat1 > &, _II2)
- template<typename _II1, typename _Seq1, typename _Cat1, typename _II2, typename _Seq2, typename _Cat2>
  constexpr bool **__equal_aux** (const ::__gnu_debug::_Safe_iterator< _II1, _Seq1, _Cat1 > &, const ::__gnu_debug::_Safe_iterator< _II1, _Seq1, _Cat1 > &, const ::__gnu_debug::_Safe_iterator< _II2, _Seq2, _Cat2 > &)
- template<typename _Tp, typename _Ref, typename _Ptr, typename _II>
  __gnu_cxx::__enable_if< __is_random_access_iter< _II >::__value, bool >::__type **__equal_aux1** (::_Deque_iterator< _Tp, _Ref, _Ptr > __first1, ::_Deque_iterator< _Tp, _Ref, _Ptr > __last1, _II __first2)
- template<typename _Tp1, typename _Ref1, typename _Ptr1, typename _Tp2, typename _Ref2, typename _Ptr2>
  bool **__equal_aux1** (::_Deque_iterator< _Tp1, _Ref1, _Ptr1 > __first1, ::_Deque_iterator< _Tp1, _Ref1, _Ptr1 > __last1, ::_Deque_iterator< _Tp2, _Ref2, _Ptr2 > __first2)
- template<typename _II, typename _Tp, typename _Ref, typename _Ptr>
  __gnu_cxx::__enable_if< __is_random_access_iter< _II >::__value, bool >::__type **__equal_aux1** (_II __first1, _II __last1, ::_Deque_iterator< _Tp, _Ref, _Ptr > __first2)

- template<typename _II1, typename _II2>
  constexpr bool **__equal_aux1** (_II1 __first1, _II1 __last1, _II2 __first2)
- template<typename _Tp, typename _Ref, typename _Ptr, typename _II>
  bool **__equal_dit** (const ::_Deque_iterator< _Tp, _Ref, _Ptr > &__first1, const ::_Deque_iterator< _Tp, _Ref, _Ptr > &__last1, _II __first2)
- template<typename _ForwardIterator, typename _Tp, typename _Compare>
  constexpr pair< _ForwardIterator, _ForwardIterator > **__equal_range** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)
- template<typename _Tp, typename _Up = _Tp>
  constexpr _Tp **__exchange** (_Tp &__obj, _Up &&__new_val)
- template<typename _FIte, typename _Tp>
  constexpr void **__fill_a** (_FIte __first, _FIte __last, const _Tp &__value)
- template<typename _Ite, typename _Seq, typename _Cat, typename _Tp>
  constexpr void **__fill_a** (const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &, const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &, const _Tp &)
- template<typename _Ite, typename _Cont, typename _Tp>
  constexpr void **__fill_a1** (::__gnu_cxx::__normal_iterator< _Ite, _Cont > __first, ::__gnu_cxx::__normal_iterator< _Ite, _Cont > __last, const _Tp &__value)
- constexpr void **__fill_a1** (::_Bit_iterator, ::_Bit_iterator, const bool &)
- template<typename _ForwardIterator, typename _Tp>
  constexpr void **__fill_a1** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)
- template<typename _Up, typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_byte< _Up >::__value &&(__are_same< _Up, _Tp >::__value||__memcpyable_integer< _Tp >::__width), void >::__type **__fill_a1** (_Up *__first, _Up *__last, const _Tp &__x)
- template<typename _Tp, typename _VTp>
  void **__fill_a1** (const ::_Deque_iterator< _Tp, _Tp &, _Tp * > &__first, const ::_Deque_iterator< _Tp, _Tp &, _Tp * > &__last, const _VTp &__value)
- constexpr void **__fill_bvector** (_Bit_type *__v, unsigned int __first, unsigned int __last, bool __x) noexcept
- constexpr void **__fill_bvector_n** (_Bit_type *, size_t, bool) noexcept
- template<typename _OutputIterator, typename _Size, typename _Tp>
  constexpr _OutputIterator **__fill_n_a** (_OutputIterator __first, _Size __n, const _Tp &__value, std::input_iterator_tag)
- template<typename _OutputIterator, typename _Size, typename _Tp>
  constexpr _OutputIterator **__fill_n_a** (_OutputIterator __first, _Size __n, const _Tp &__value, std::output_iterator_tag)
- template<typename _OutputIterator, typename _Size, typename _Tp>
  constexpr _OutputIterator **__fill_n_a** (_OutputIterator __first, _Size __n, const _Tp &__value, std::random_access_iterator_tag)
- template<typename _Ite, typename _Seq, typename _Cat, typename _Size, typename _Tp>
  constexpr ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > **__fill_n_a** (const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &__first, _Size __n, const _Tp &__value, std::input_iterator_tag)
- template<typename _OutputIterator, typename _Size, typename _Tp>
  constexpr _OutputIterator **__fill_n_a1** (_OutputIterator __first, _Size __n, const _Tp &__value)
- template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _BinaryPredicate>
  constexpr _BidirectionalIterator1 **__find_end** (_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, bidirectional_iterator_tag, bidirectional_iterator_tag, _BinaryPredicate __comp)
- template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>
  constexpr _ForwardIterator1 **__find_end** (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, forward_iterator_tag, forward_iterator_tag, _BinaryPredicate __comp)
- template<typename _Iterator, typename _Predicate>
  constexpr _Iterator **__find_if** (_Iterator __first, _Iterator __last, _Predicate __pred)
- template<typename _InputIterator, typename _Predicate>
  constexpr _InputIterator __find_if_not (_InputIterator __first, _InputIterator __last, _Predicate __pred)

- template<typename _InputIterator, typename _Predicate, typename _Distance>
  constexpr _InputIterator __find_if_not_n (_InputIterator __first, _Distance &__len, _Predicate __pred)
- template<typename _Tp, typename... _Types>
  constexpr size_t **__find_uniq_type_in_pack** ()
- from_chars_result **__from_chars_bfloat16_t** (const char *__first, const char *__last, float &__value, chars_format __fmt=chars_format::general) noexcept
- from_chars_result **__from_chars_float16_t** (const char *__first, const char *__last, float &__value, chars_format __fmt=chars_format::general) noexcept
- template<typename _EuclideanRingElement>
  constexpr _EuclideanRingElement __gcd (_EuclideanRingElement __m, _EuclideanRingElement __n)
- template<typename _IntType, typename _UniformRandomBitGenerator>
  pair< _IntType, _IntType > __gen_two_uniform_ints (_IntType __b0, _IntType __b1, _UniformRandomBit↩ Generator &&__g)
- template<size_t __i, typename _Head, typename... _Tail>
  constexpr _Head & **__get_helper** (_Tuple_impl< __i, _Head, _Tail... > &__t) noexcept
- template<size_t __i, typename _Head, typename... _Tail>
  constexpr const _Head & **__get_helper** (const _Tuple_impl< __i, _Head, _Tail... > &__t) noexcept
- template<size_t __i, typename... _Types>
  __enable_if_t<(__i >=sizeof...(_Types))> **__get_helper** (const tuple< _Types... > &)=delete
- void **__glibcxx_assert_fail** (const char *__file, int __line, const char *__function, const char *__condition) noexcept
- template<typename _Tp>
  size_t **__iconv_adaptor** (size_t(*__func)(iconv_t, _Tp, size_t *, char **, size_t *), iconv_t __cd, char **__inbuf, size_t *__inbytes, char **__outbuf, size_t *__outbytes)
- template<typename _InputIterator1, typename _InputIterator2, typename _Compare>
  constexpr bool **__includes** (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Input↩ Iterator2 __last2, _Compare __comp)
- template<typename _BidirectionalIterator, typename _Compare>
  void **__inplace_merge** (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Compare __comp)
- template<typename _RandomAccessIterator, typename _Compare>
  void __inplace_stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _CharT, typename _ValueT>
  int **__int_to_char** (_CharT *__bufend, _ValueT __v, const _CharT *__lit, ios_base::fmtflags __flags, bool __dec)
- template<typename _Tp, typename _Up = typename __inv_unwrap<_Tp>::type>
  constexpr _Up && **__invfwd** (typename remove_reference< _Tp >::type &__t) noexcept
- template<typename _Callable, typename... _Args>
  constexpr __invoke_result< _Callable, _Args... >::type __invoke (_Callable &&__fn, _Args &&... __args) noexcept(__is_nothrow_invocable< _Callable, _Args... >::value)
- template<typename _Res, typename _MemFun, typename _Tp, typename... _Args>
  constexpr _Res **__invoke_impl** (__invoke_memfun_deref, _MemFun &&__f, _Tp &&__t, _Args &&... __args)
- template<typename _Res, typename _MemFun, typename _Tp, typename... _Args>
  constexpr _Res **__invoke_impl** (__invoke_memfun_ref, _MemFun &&__f, _Tp &&__t, _Args &&... __args)
- template<typename _Res, typename _MemPtr, typename _Tp>
  constexpr _Res **__invoke_impl** (__invoke_memobj_deref, _MemPtr &&__f, _Tp &&__t)
- template<typename _Res, typename _MemPtr, typename _Tp>
  constexpr _Res **__invoke_impl** (__invoke_memobj_ref, _MemPtr &&__f, _Tp &&__t)
- template<typename _Res, typename _Fn, typename... _Args>
  constexpr _Res **__invoke_impl** (__invoke_other, _Fn &&__f, _Args &&... __args)
- template<typename _Res, typename _Callable, typename... _Args>
  constexpr enable_if_t< is_invocable_r_v< _Res, _Callable, _Args... >, _Res > **__invoke_r** (_Callable &&__fn, _Args &&... __args) noexcept(is_nothrow_invocable_r_v< _Res, _Callable, _Args... >)
- constexpr bool **__is_constant_evaluated** () noexcept

- template<typename _RandomAccessIterator, typename _Compare, typename _Distance>
  constexpr bool **__is_heap** (_RandomAccessIterator __first, _Compare __comp, _Distance __n)
- template<typename _RandomAccessIterator, typename _Distance>
  constexpr bool **__is_heap** (_RandomAccessIterator __first, _Distance __n)
- template<typename _RandomAccessIterator>
  constexpr bool **__is_heap** (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr bool **__is_heap** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _RandomAccessIterator, typename _Distance, typename _Compare>
  constexpr _Distance **__is_heap_until** (_RandomAccessIterator __first, _Distance __n, _Compare &__comp)
- template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>
  constexpr bool **__is_permutation** (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 _↩
  _first2, _BinaryPredicate __pred)
- template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>
  constexpr bool **__is_permutation** (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 _↩
  _first2, _ForwardIterator2 __last2, _BinaryPredicate __pred)
- template<typename _ForwardIterator, typename _Compare>
  constexpr _ForwardIterator **__is_sorted_until** (_ForwardIterator __first, _ForwardIterator __last, _Compare _↩
  _comp)
- template<typename _CharT, typename _Traits>
  void **__istream_extract** (basic_istream< _CharT, _Traits > &, _CharT *, streamsize)
- void **__istream_extract** (istream &, char *, streamsize)
- template<typename _Iter>
  constexpr iterator_traits< _Iter >::iterator_category __iterator_category (const _Iter &)
- template<typename _Tp1, typename _Ref, typename _Ptr, typename _Tp2>
  int **__lex_cmp_dit** (::_Deque_iterator< _Tp1, _Ref, _Ptr > __first1, ::_Deque_iterator< _Tp1, _Ref, _Ptr > _↩
  _last1, const _Tp2 *__first2, const _Tp2 *__last2)
- template<typename _II1, typename _II2>
  constexpr bool **__lexicographical_compare_aux** (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)
- template<typename _II1, typename _Ite2, typename _Seq2, typename _Cat2>
  constexpr bool **__lexicographical_compare_aux** (_II1 __first1, _II1 __last1, const ::__gnu_debug::_Safe_iterator<
  _Ite2, _Seq2, _Cat2 > &__first2, const ::__gnu_debug::_Safe_iterator< _Ite2, _Seq2, _Cat2 > &__last2)
- template<typename _II1, typename _Iter2, typename _Seq2, typename _Cat2>
  constexpr bool **__lexicographical_compare_aux** (_II1, _II1, const ::__gnu_debug::_Safe_iterator< _Iter2, _↩
  Seq2, _Cat2 > &, const ::__gnu_debug::_Safe_iterator< _Iter2, _Seq2, _Cat2 > &)
- template<typename _Ite1, typename _Seq1, typename _Cat1, typename _II2>
  constexpr bool **__lexicographical_compare_aux** (const ::__gnu_debug::_Safe_iterator< _Ite1, _Seq1, _Cat1
  > &__first1, const ::__gnu_debug::_Safe_iterator< _Ite1, _Seq1, _Cat1 > &__last1, _II2 __first2, _II2 __last2)
- template<typename _Ite1, typename _Seq1, typename _Cat1, typename _Ite2, typename _Seq2, typename _Cat2>
  constexpr bool **__lexicographical_compare_aux** (const ::__gnu_debug::_Safe_iterator< _Ite1, _Seq1, ↩
  _Cat1 > &__first1, const ::__gnu_debug::_Safe_iterator< _Ite1, _Seq1, _Cat1 > &__last1, const
  ::__gnu_debug::_Safe_iterator< _Ite2, _Seq2, _Cat2 > &__first2, const ::__gnu_debug::_Safe_iterator< ↩
  _Ite2, _Seq2, _Cat2 > &__last2)
- template<typename _Iter1, typename _Seq1, typename _Cat1, typename _II2>
  constexpr bool **__lexicographical_compare_aux** (const ::__gnu_debug::_Safe_iterator< _Iter1, _Seq1, _Cat1
  > &, const ::__gnu_debug::_Safe_iterator< _Iter1, _Seq1, _Cat1 > &, _II2, _II2)
- template<typename _Iter1, typename _Seq1, typename _Cat1, typename _Iter2, typename _Seq2, typename _Cat2>
  constexpr bool **__lexicographical_compare_aux** (const ::__gnu_debug::_Safe_iterator< _Iter1, _Seq1, _Cat1
  > &, const ::__gnu_debug::_Safe_iterator< _Iter1, _Seq1, _Cat1 > &, const ::__gnu_debug::_Safe_iterator<
  _Iter2, _Seq2, _Cat2 > &, const ::__gnu_debug::_Safe_iterator< _Iter2, _Seq2, _Cat2 > &)
- template<typename _Tp1, typename _Ref1, typename _Ptr1, typename _Tp2, typename _Ref2, typename _Ptr2>
  bool **__lexicographical_compare_aux1** (::_Deque_iterator< _Tp1, _Ref1, _Ptr1 > __first1, ::_Deque_iterator<
  _Tp1, _Ref1, _Ptr1 > __last1, ::_Deque_iterator< _Tp2, _Ref2, _Ptr2 > __first2, ::_Deque_iterator< _Tp2,
  _Ref2, _Ptr2 > __last2)

- template<typename _Tp1, typename _Ref1, typename _Ptr1, typename _Tp2>
  bool **__lexicographical_compare_aux1** (::_Deque_iterator< _Tp1, _Ref1, _Ptr1 > __first1, ::_Deque_iterator< _Tp1, _Ref1, _Ptr1 > __last1, _Tp2 *__first2, _Tp2 *__last2)
- template<typename _II1, typename _II2>
  constexpr bool **__lexicographical_compare_aux1** (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)
- template<typename _Tp1, typename _Tp2, typename _Ref2, typename _Ptr2>
  bool **__lexicographical_compare_aux1** (_Tp1 *__first1, _Tp1 *__last1, ::_Deque_iterator< _Tp2, _Ref2, _Ptr2 > __first2, ::_Deque_iterator< _Tp2, _Ref2, _Ptr2 > __last2)
- template<typename _II1, typename _II2, typename _Compare>
  constexpr bool **__lexicographical_compare_impl** (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, _Compare __comp)
- template<typename _Tp>
  constexpr _Tp __lg (_Tp __n)
- template<typename _ForwardIterator, typename _Tp, typename _Compare>
  constexpr _ForwardIterator **__lower_bound** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)
- template<typename... _BoundArgs, typename... _Args>
  constexpr auto **__make_bound_args** (_Args &&... __args)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void **__make_heap** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare &_comp)
- template<typename _Iterator, typename _ReturnType = __conditional_t<__move_if_noexcept_cond <typename iterator_traits<_Iterator>::value_type>::value, _Iterator, move_iterator<_Iterator>>>
  constexpr _ReturnType **__make_move_if_noexcept_iterator** (_Iterator __i)
- template<typename _Tp, typename _ReturnType = __conditional_t<__move_if_noexcept_cond<_Tp>::value, const _Tp*, move_iterator<_Tp*>>>
  constexpr _ReturnType **__make_move_if_noexcept_iterator** (_Tp *__i)
- template<typename _Iterator>
  constexpr reverse_iterator< _Iterator > **__make_reverse_iterator** (_Iterator __i)
- template<typename _Tp, _Lock_policy _Lp = __default_lock_policy, typename... _Args>
  __shared_ptr< _Tp, _Lp > **__make_shared** (_Args &&... __args)
- template<typename _ForwardIterator, typename _Compare>
  constexpr _ForwardIterator **__max_element** (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)
- template<typename _Tp, typename _Up>
  constexpr int **__memcmp** (const _Tp *__first1, const _Up *__first2, size_t __num)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator **__merge** (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _BidirectionalIterator, typename _Distance, typename _Pointer, typename _Compare>
  void __merge_adaptive (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Pointer __buffer, _Compare __comp)
- template<typename _BidirectionalIterator, typename _Distance, typename _Pointer, typename _Compare>
  void **__merge_adaptive_resize** (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Pointer __buffer, _Distance __buffer_size, _Compare __comp)
- template<typename _RandomAccessIterator1, typename _RandomAccessIterator2, typename _Distance, typename _Compare>
  void **__merge_sort_loop** (_RandomAccessIterator1 __first, _RandomAccessIterator1 __last, _RandomAccessIterator2 __result, _Distance __step_size, _Compare __comp)
- template<typename _RandomAccessIterator, typename _Pointer, typename _Compare>
  void **__merge_sort_with_buffer** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __buffer, _Compare __comp)

- template<typename _BidirectionalIterator, typename _Distance, typename _Compare>
  void __merge_without_buffer (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Compare __comp)
- template<typename _Tp>
  constexpr auto **__min_cmp** (_Tp __x, _Tp __y)
- template<typename _ForwardIterator, typename _Compare>
  constexpr _ForwardIterator **__min_element** (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)
- template<typename _ForwardIterator, typename _Compare>
  constexpr pair< _ForwardIterator, _ForwardIterator > **__minmax_element** (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)
- template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>
  constexpr pair< _InputIterator1, _InputIterator2 > **__mismatch** (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_pred)
- template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>
  constexpr pair< _InputIterator1, _InputIterator2 > **__mismatch** (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _BinaryPredicate __binary_pred)
- template<typename _Iterator>
  constexpr _Iterator **__miter_base** (_Iterator __it)
- template<typename _Iterator, typename _Compare>
  constexpr void __move_median_to_first (_Iterator __result, _Iterator __a, _Iterator __b, _Iterator __c, _Compare __comp)
- template<typename _InputIterator, typename _OutputIterator, typename _Compare>
  _OutputIterator __move_merge (_InputIterator __first1, _InputIterator __last1, _InputIterator __first2, _InputIterator __last2, _OutputIterator __result, _Compare __comp)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  void __move_merge_adaptive (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _BidirectionalIterator3, typename _Compare>
  void __move_merge_adaptive_backward (_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, _BidirectionalIterator3 __result, _Compare __comp)
- template<typename _BidirectionalIterator, typename _Compare>
  constexpr bool **__next_permutation** (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)
- template<typename _Ite, typename _Seq>
  constexpr decltype(std::__niter_base(std::declval< _Ite >())) **__niter_base** (const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, std::random_access_iterator_tag > &__it) noexcept(std::is_nothrow_copy_constructible< _Ite >::value)
- void **__once_proxy** (void)
- template<typename _BidirectionalIterator, typename _Predicate>
  constexpr _BidirectionalIterator __partition (_BidirectionalIterator __first, _BidirectionalIterator __last, _Predicate __pred, bidirectional_iterator_tag)
- template<typename _ForwardIterator, typename _Predicate>
  constexpr _ForwardIterator __partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, forward_iterator_tag)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void **__pop_heap** (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomAccessIterator __result, _Compare &__comp)
- template<typename _BidirectionalIterator, typename _Compare>
  constexpr bool **__prev_permutation** (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)
- template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _Compare>
  constexpr void **__push_heap** (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __topIndex, _Tp __value, _Compare &__comp)

- template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type>
  _Out_iter **__regex_replace** (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _↩ Rx_traits > &__e, const _Ch_type *__fmt, size_t __len, regex_constants::match_flag_type __flags)
- template<typename _InputIterator, typename _OutputIterator, typename _Predicate>
  constexpr _OutputIterator **__remove_copy_if** (_InputIterator __first, _InputIterator __last, _OutputIterator __↩ result, _Predicate __pred)
- template<typename _ForwardIterator, typename _Predicate>
  constexpr _ForwardIterator **__remove_if** (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)
- template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _Tp>
  constexpr _OutputIterator **__replace_copy_if** (_InputIterator __first, _InputIterator __last, _OutputIterator __↩ result, _Predicate __pred, const _Tp &__new_value)
- template<typename _BidirectionalIterator>
  constexpr void __reverse (_BidirectionalIterator __first, _BidirectionalIterator __last, bidirectional_iterator_tag)
- template<typename _RandomAccessIterator>
  constexpr void __reverse (_RandomAccessIterator __first, _RandomAccessIterator __last, random_access_iterator_tag)
- template<typename _BidirectionalIterator>
  constexpr _BidirectionalIterator __rotate (_BidirectionalIterator __first, _BidirectionalIterator __middle, _↩ BidirectionalIterator __last, bidirectional_iterator_tag)
- template<typename _ForwardIterator>
  constexpr _ForwardIterator __rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __↩ last, forward_iterator_tag)
- template<typename _RandomAccessIterator>
  constexpr _RandomAccessIterator __rotate (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, random_access_iterator_tag)
- template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _Distance>
  _BidirectionalIterator1 __rotate_adaptive (_BidirectionalIterator1 __first, _BidirectionalIterator1 __middle, _↩ BidirectionalIterator1 __last, _Distance __len1, _Distance __len2, _BidirectionalIterator2 __buffer, _Distance _↩ _buffer_size)
- template<typename _ForwardIterator, typename _OutputIterator, typename _Cat, typename _Size, typename _UniformRandomBit↩ Generator>
  _OutputIterator __sample (_ForwardIterator __first, _ForwardIterator __last, forward_iterator_tag, _OutputIterator __out, _Cat, _Size __n, _UniformRandomBitGenerator &&__g)
- template<typename _InputIterator, typename _RandomAccessIterator, typename _Size, typename _UniformRandomBitGenerator>
  _RandomAccessIterator __sample (_InputIterator __first, _InputIterator __last, input_iterator_tag, _Random↩ AccessIterator __out, random_access_iterator_tag, _Size __n, _UniformRandomBitGenerator &&__g)
- template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>
  constexpr _ForwardIterator1 **__search** (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __predicate)
- template<typename _ForwardIterator, typename _Integer, typename _UnaryPredicate>
  constexpr _ForwardIterator **__search_n** (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, _↩ UnaryPredicate __unary_pred)
- template<typename _ForwardIterator, typename _Integer, typename _UnaryPredicate>
  constexpr _ForwardIterator __search_n_aux (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, _UnaryPredicate __unary_pred, std::forward_iterator_tag)
- template<typename _RandomAccessIter, typename _Integer, typename _UnaryPredicate>
  constexpr _RandomAccessIter __search_n_aux (_RandomAccessIter __first, _RandomAccessIter __last, _↩ Integer __count, _UnaryPredicate __unary_pred, std::random_access_iterator_tag)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator **__set_difference** (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 _↩ _first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator **__set_intersection** (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)

- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator **__set_symmetric_difference** (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator **__set_union** (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _Tp>
  _Tp * **__shared_ptr_deref** (_Tp *__p)
- constexpr long long **__size_to_integer** (double __n)
- constexpr long long **__size_to_integer** (float __n)
- constexpr int **__size_to_integer** (int __n)
- constexpr long **__size_to_integer** (long __n)
- constexpr long long **__size_to_integer** (long double __n)
- constexpr long long **__size_to_integer** (long long __n)
- constexpr unsigned **__size_to_integer** (unsigned __n)
- constexpr unsigned long **__size_to_integer** (unsigned long __n)
- constexpr unsigned long long **__size_to_integer** (unsigned long long __n)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void **__sort_heap** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare &__comp)
- template<typename _ForwardIterator, typename _Predicate>
  _ForwardIterator **__stable_partition** (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)
- template<typename _ForwardIterator, typename _Pointer, typename _Predicate, typename _Distance>
  _ForwardIterator __stable_partition_adaptive (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, _Distance __len, _Pointer __buffer, _Distance __buffer_size)
- template<typename _RandomAccessIterator, typename _Compare>
  void **__stable_sort** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _RandomAccessIterator, typename _Pointer, typename _Compare>
  void **__stable_sort_adaptive** (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Pointer __buffer, _Compare __comp)
- template<typename _RandomAccessIterator, typename _Pointer, typename _Distance, typename _Compare>
  void **__stable_sort_adaptive_resize** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __buffer, _Distance __buffer_size, _Compare __comp)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _State>
  bool **__str_codecvt_in** (const char *__first, const char *__last, basic_string< _CharT, _Traits, _Alloc > &__outstr, const codecvt< _CharT, char, _State > &__cvt)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _State>
  bool **__str_codecvt_in** (const char *__first, const char *__last, basic_string< _CharT, _Traits, _Alloc > &__outstr, const codecvt< _CharT, char, _State > &__cvt, _State &__state, size_t &__count)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _State>
  bool **__str_codecvt_in_all** (const char *__first, const char *__last, basic_string< _CharT, _Traits, _Alloc > &__outstr, const codecvt< _CharT, char, _State > &__cvt)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _State>
  bool **__str_codecvt_out** (const _CharT *__first, const _CharT *__last, basic_string< char, _Traits, _Alloc > &__outstr, const codecvt< _CharT, char, _State > &__cvt)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _State>
  bool **__str_codecvt_out** (const _CharT *__first, const _CharT *__last, basic_string< char, _Traits, _Alloc > &__outstr, const codecvt< _CharT, char, _State > &__cvt, _State &__state, size_t &__count)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _State>
  bool **__str_codecvt_out_all** (const _CharT *__first, const _CharT *__last, basic_string< char, _Traits, _Alloc > &__outstr, const codecvt< _CharT, char, _State > &__cvt)

- template<typename _Str>
  constexpr _Str **__str_concat** (typename _Str::value_type const *__lhs, typename _Str::size_type __lhs_len, typename _Str::value_type const *__rhs, typename _Str::size_type __rhs_len, typename _Str::allocator_type const &__a)
- constexpr size_t **__sv_check** (size_t __size, size_t __pos, const char *__s)
- constexpr size_t **__sv_limit** (size_t __size, size_t __pos, size_t __off) noexcept
- void **__terminate** () noexcept
- void **__throw_bad_alloc** (void)
- void **__throw_bad_array_new_length** (void)
- void **__throw_bad_cast** (void)
- void **__throw_bad_exception** (void)
- void **__throw_bad_function_call** ()
- void **__throw_bad_typeid** (void)
- void **__throw_bad_weak_ptr** ()
- void **__throw_domain_error** (const char *)
- void **__throw_future_error** (int)
- void **__throw_invalid_argument** (const char *)
- void **__throw_ios_failure** (const char *)
- void **__throw_ios_failure** (const char *, int)
- void **__throw_length_error** (const char *)
- void **__throw_logic_error** (const char *)
- void **__throw_out_of_range** (const char *)
- void **__throw_out_of_range_fmt** (const char *,...)
- void **__throw_overflow_error** (const char *)
- void **__throw_range_error** (const char *)
- void **__throw_runtime_error** (const char *)
- void **__throw_system_error** (int)
- void **__throw_underflow_error** (const char *)
- template<typename _Tp>
  constexpr to_chars_result **__to_chars_i** (char *__first, char *__last, _Tp __value, int __base=10)
- constexpr void **__to_wstring_numeric** (const char *__s, int __len, wchar_t *__wout)
- wstring **__to_wstring_numeric** (const string &__s)
- template<typename _Facet>
  const _Facet * **__try_use_facet** (const locale &__loc)
- template<typename _Cat, typename _Tp, typename _Up, typename _IndexSeq>
  constexpr _Cat **__tuple_cmp** (const _Tp &__t, const _Up &__u, _IndexSeq __indices)
- template<typename _Tp, typename _ForwardIterator>
  void **__uninitialized_construct_buf** (_Tp *__first, _Tp *__last, _ForwardIterator __seed)
- template<typename _ForwardIterator, typename _BinaryPredicate>
  constexpr _ForwardIterator **__unique** (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)
- template<typename _ForwardIterator, typename _OutputIterator, typename _BinaryPredicate>
  constexpr _OutputIterator **__unique_copy** (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator _result, _BinaryPredicate __binary_pred, forward_iterator_tag)
- template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate>
  constexpr _OutputIterator **__unique_copy** (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred, input_iterator_tag)
- template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate>
  _ForwardIterator **__unique_copy_1** (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, BinaryPredicate __binary_pred, __true_type)

- template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate>
  constexpr _OutputIterator **__unique_copy_1** (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred, __false_type)
- template<typename _ForwardIterator, typename _Tp, typename _Compare>
  constexpr _ForwardIterator **__upper_bound** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)
- template<typename _Tp>
  void **__valarray_copy** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **__valarray_copy** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp>
  void **__valarray_copy** (_Array< _Tp > __a, _Array< bool > __m, size_t __n, _Array< _Tp > __b, _Array< bool > __k)
- template<typename _Tp>
  void **__valarray_copy** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp>
  void **__valarray_copy** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp>
  void **__valarray_copy** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp>
  void **__valarray_copy** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp>
  void **__valarray_copy** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp>
  void **__valarray_copy** (_Array< _Tp > __a, size_t __n, size_t __s1, _Array< _Tp > __b, size_t __s2)
- template<typename _Tp>
  void **__valarray_copy** (_Array< _Tp > __e, _Array< size_t > __f, size_t __n, _Array< _Tp > __a, _Array< size_t > __i)
- template<typename _Tp>
  void **__valarray_copy** (_Array< _Tp > __src, size_t __n, _Array< size_t > __i, _Array< _Tp > __dst, _Array< size_t > __j)
- template<typename _Tp, class _Dom>
  void **__valarray_copy** (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a)
- template<typename _Tp, class _Dom>
  void **__valarray_copy** (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, _Array< bool > __m)
- template<typename _Tp, class _Dom>
  void **__valarray_copy** (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, _Array< size_t > __i)
- template<typename _Tp, class _Dom>
  void **__valarray_copy** (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, size_t __s)
- template<typename _Tp>
  void **__valarray_copy** (const _Tp *__restrict__ __a, _Tp *__restrict__ __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **__valarray_copy** (const _Tp *__restrict__ __a, const size_t *__restrict__ __i, _Tp *__restrict__ __b, size_t __n)
- template<typename _Tp>
  void **__valarray_copy** (const _Tp *__restrict__ __a, size_t __n, _Tp *__restrict__ __b)
- template<typename _Tp>
  void **__valarray_copy** (const _Tp *__restrict__ __a, size_t __n, _Tp *__restrict__ __b, const size_t *__restrict__ __i)
- template<typename _Tp>
  void **__valarray_copy** (const _Tp *__restrict__ __a, size_t __n, size_t __s, _Tp *__restrict__ __b)

- template<typename _Tp>
  void **__valarray_copy** (const _Tp *__restrict__ __src, size_t __n, const size_t *__restrict__ __i, _Tp *__restrict__ __dst, const size_t *__restrict__ __j)
- template<typename _Tp>
  void **__valarray_copy** (const _Tp *__restrict__ __src, size_t __n, size_t __s1, _Tp *__restrict__ __dst, size_t __s2)
- template<typename _Tp>
  void **__valarray_copy_construct** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp>
  void **__valarray_copy_construct** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp>
  void **__valarray_copy_construct** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp, class _Dom>
  void **__valarray_copy_construct** (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a)
- template<typename _Tp>
  void **__valarray_copy_construct** (const _Tp *__b, const _Tp *__e, _Tp *__restrict__ __o)
- template<typename _Tp>
  void **__valarray_copy_construct** (const _Tp *__restrict__ __a, const size_t *__restrict__ __i, _Tp *__restrict__ __o, size_t __n)
- template<typename _Tp>
  void **__valarray_copy_construct** (const _Tp *__restrict__ __a, size_t __n, size_t __s, _Tp *__restrict__ __o)
- template<typename _Tp>
  void **__valarray_default_construct** (_Tp *__b, _Tp *__e)
- template<typename _Tp>
  void **__valarray_destroy_elements** (_Tp *__b, _Tp *__e)
- template<typename _Tp>
  void **__valarray_fill** (_Array< _Tp > __a, _Array< size_t > __i, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **__valarray_fill** (_Array< _Tp > __a, size_t __n, _Array< bool > __m, const _Tp &__t)
- template<typename _Tp>
  void **__valarray_fill** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **__valarray_fill** (_Array< _Tp > __a, size_t __n, size_t __s, const _Tp &__t)
- template<typename _Tp>
  void **__valarray_fill** (_Tp *__restrict__ __a, const size_t *__restrict__ __i, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **__valarray_fill** (_Tp *__restrict__ __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **__valarray_fill** (_Tp *__restrict__ __a, size_t __n, size_t __s, const _Tp &__t)
- template<typename _Tp>
  void **__valarray_fill_construct** (_Tp *__b, _Tp *__e, const _Tp __t)
- template<typename _Tp>
  _Tp * **__valarray_get_storage** (size_t)
- template<typename _Ta>
  _Ta::value_type **__valarray_max** (const _Ta &__a)
- template<typename _Ta>
  _Ta::value_type **__valarray_min** (const _Ta &__a)
- void **__valarray_release_memory** (void *__p)
- template<typename _Tp>
  _Tp **__valarray_sum** (const _Tp *__f, const _Tp *__l)
- template<template< typename > class _Trait, typename _Tp, typename _Up = _Tp>
  constexpr _Up **__value_or** (_Up __def=_Up()) noexcept

- bool **__verify_grouping** (const char *__grouping, size_t __grouping_size, const string &__grouping_tmp) throw ()
- template<typename _CharT, typename _OutIter>
  _OutIter **__write** (_OutIter __s, const _CharT *__ws, int __len)
- template<typename _CharT>
  ostreambuf_iterator< _CharT > **__write** (ostreambuf_iterator< _CharT > __s, const _CharT *__ws, int __len)
- template<typename _Tp>
  void **_Array_augmented___bitwise_and** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **_Array_augmented___bitwise_and** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___bitwise_and** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___bitwise_and** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___bitwise_and** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___bitwise_and** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___bitwise_and** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp>
  void **_Array_augmented___bitwise_and** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp>
  void **_Array_augmented___bitwise_and** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp>
  void **_Array_augmented___bitwise_and** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **_Array_augmented___bitwise_and** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___bitwise_and** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___bitwise_or** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **_Array_augmented___bitwise_or** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___bitwise_or** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___bitwise_or** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___bitwise_or** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___bitwise_or** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)

- template<typename _Tp>
  void **_Array_augmented___bitwise_or** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp>
  void **_Array_augmented___bitwise_or** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp>
  void **_Array_augmented___bitwise_or** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp>
  void **_Array_augmented___bitwise_or** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **_Array_augmented___bitwise_or** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___bitwise_or** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___bitwise_xor** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **_Array_augmented___bitwise_xor** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___bitwise_xor** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___bitwise_xor** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___bitwise_xor** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___bitwise_xor** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___bitwise_xor** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp>
  void **_Array_augmented___bitwise_xor** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp>
  void **_Array_augmented___bitwise_xor** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp>
  void **_Array_augmented___bitwise_xor** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **_Array_augmented___bitwise_xor** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___bitwise_xor** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___divides** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **_Array_augmented___divides** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)

- template<typename _Tp, class _Dom>
  void **_Array_augmented___divides** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___divides** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___divides** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___divides** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___divides** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp>
  void **_Array_augmented___divides** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp>
  void **_Array_augmented___divides** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp>
  void **_Array_augmented___divides** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **_Array_augmented___divides** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___divides** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___minus** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **_Array_augmented___minus** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___minus** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___minus** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___minus** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___minus** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___minus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp>
  void **_Array_augmented___minus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp>
  void **_Array_augmented___minus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp>
  void **_Array_augmented___minus** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **_Array_augmented___minus** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___minus** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)

- template<typename _Tp>
  void **_Array_augmented___modulus** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **_Array_augmented___modulus** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___modulus** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___modulus** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___modulus** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___modulus** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___modulus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp>
  void **_Array_augmented___modulus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp>
  void **_Array_augmented___modulus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp>
  void **_Array_augmented___modulus** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **_Array_augmented___modulus** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___modulus** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___multiplies** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **_Array_augmented___multiplies** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___multiplies** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___multiplies** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___multiplies** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___multiplies** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___multiplies** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp>
  void **_Array_augmented___multiplies** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)

- template<typename _Tp>
  void **_Array_augmented___multiplies** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp>
  void **_Array_augmented___multiplies** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **_Array_augmented___multiplies** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___multiplies** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___plus** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **_Array_augmented___plus** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___plus** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___plus** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___plus** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___plus** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___plus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp>
  void **_Array_augmented___plus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp>
  void **_Array_augmented___plus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp>
  void **_Array_augmented___plus** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **_Array_augmented___plus** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___plus** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___shift_left** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **_Array_augmented___shift_left** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___shift_left** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___shift_left** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___shift_left** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___shift_left** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)

- template<typename _Tp>
  void **_Array_augmented___shift_left** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp>
  void **_Array_augmented___shift_left** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp>
  void **_Array_augmented___shift_left** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp>
  void **_Array_augmented___shift_left** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **_Array_augmented___shift_left** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___shift_left** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___shift_right** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **_Array_augmented___shift_right** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___shift_right** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___shift_right** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___shift_right** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___shift_right** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___shift_right** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp>
  void **_Array_augmented___shift_right** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp>
  void **_Array_augmented___shift_right** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp>
  void **_Array_augmented___shift_right** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **_Array_augmented___shift_right** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___shift_right** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp, typename... _Args>
  constexpr void _Construct (_Tp *__p, _Args &&... __args)
- template<typename _T1>
  void **_Construct_novalue** (_T1 *__p)
- template<typename _ForwardIterator>
  constexpr void _Destroy (_ForwardIterator __first, _ForwardIterator __last)

- template<typename _Tp>
  constexpr void _Destroy (_Tp *__pointer)
- template<typename _ForwardIterator, typename _Size>
  constexpr _ForwardIterator _Destroy_n (_ForwardIterator __first, _Size __count)
- size_t **_Fnv_hash_bytes** (const void *__ptr, size_t __len, size_t __seed)
- size_t **_Hash_bytes** (const void *__ptr, size_t __len, size_t __seed)
- unsigned int **_Rb_tree_black_count** (const _Rb_tree_node_base *__node, const _Rb_tree_node_base *__root) throw ()
- _Rb_tree_node_base * **_Rb_tree_decrement** (_Rb_tree_node_base *__x) throw ()
- _Rb_tree_node_base * **_Rb_tree_increment** (_Rb_tree_node_base *__x) throw ()
- void **_Rb_tree_insert_and_rebalance** (const bool __insert_left, _Rb_tree_node_base *__x, _Rb_tree_node_base *__p, _Rb_tree_node_base &__header) throw ()
- _Rb_tree_node_base * **_Rb_tree_rebalance_for_erase** (_Rb_tree_node_base *const __z, _Rb_tree_node_base &__header) throw ()
- template<class _Dom>
  _Expr< _UnClos< struct std::_Abs, _Expr, _Dom >, typename _Dom::value_type > **abs** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  _Tp abs (const complex< _Tp > &)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Abs, _ValArray, _Tp >, _Tp > **abs** (const valarray< _Tp > &__v)
- constexpr double **abs** (double __x)
- constexpr float **abs** (float __x)
- long **abs** (long __i)
- constexpr long double **abs** (long double __x)
- long long **abs** (long long __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::abs(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **abs** (simd< _Tp, _Abi > __x)
- template<typename _InputIterator, typename _Tp>
  constexpr _Tp accumulate (_InputIterator __first, _InputIterator __last, _Tp __init)
- template<typename _InputIterator, typename _Tp, typename _BinaryOperation>
  constexpr _Tp accumulate (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __binary_op)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **acos** (_Tp __x)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Acos, _Expr, _Dom >, typename _Dom::value_type > **acos** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  std::complex< _Tp > **acos** (const std::complex< _Tp > &)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Acos, _ValArray, _Tp >, _Tp > **acos** (const valarray< _Tp > &__v)
- constexpr float **acos** (float __x)
- constexpr long double **acos** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::acos(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **acos** (simd< _Tp, _Abi > __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **acosh** (_Tp __x)

- template<typename _Tp>
  std::complex< _Tp > **acosh** (const std::complex< _Tp > &)
- constexpr float acosh (float __x)
- constexpr long double **acosh** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::acosh(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **acosh** (simd< _Tp, _Abi > __x)
- template<typename _Tp>
  constexpr _Tp * addressof (_Tp &__r) noexcept
- template<typename _Tp>
  const _Tp * **addressof** (const _Tp &&)=delete
- template<typename _InputIterator, typename _OutputIterator>
  constexpr _OutputIterator adjacent_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __result)
- template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation>
  constexpr _OutputIterator adjacent_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op)
- template<typename _FIter>
  constexpr _FIter **adjacent_find** (_FIter, _FIter)
- template<typename _FIter, typename _BinaryPredicate>
  constexpr _FIter **adjacent_find** (_FIter, _FIter, _BinaryPredicate)
- template<typename _ForwardIterator>
  constexpr _ForwardIterator adjacent_find (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _BinaryPredicate>
  constexpr _ForwardIterator adjacent_find (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)
- template<typename _InputIterator, typename _Distance>
  constexpr void advance (_InputIterator &__i, _Distance __n)
- template<typename _CharT, typename _Distance>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, void >::__type **advance** (istreambuf_iterator< _CharT > &__i, _Distance __n)
- void * align (size_t __align, size_t __size, void *&__ptr, size_t &__space) noexcept
- template<typename _IIter, typename _Predicate>
  constexpr bool **all_of** (_IIter, _IIter, _Predicate)
- template<typename _InputIterator, typename _Predicate>
  constexpr bool all_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _IIter, typename _Predicate>
  constexpr bool **any_of** (_IIter, _IIter, _Predicate)
- template<typename _InputIterator, typename _Predicate>
  constexpr bool any_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type **arg** (_Tp __x)
- template<typename _Tp>
  _Tp arg (const complex< _Tp > &)
- template<typename _Tp, typename... _Up>
  array (_Tp, _Up...) -> array< enable_if_t<(is_same_v< _Tp, _Up > &&...), _Tp >, 1+sizeof...(_Up)>
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **asin** (_Tp __x)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Asin, _Expr, _Dom >, typename _Dom::value_type > **asin** (const _Expr< _Dom, typename _Dom::value_type > &__e)

- template<typename _Tp>
  std::complex< _Tp > **asin** (const std::complex< _Tp > &)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Asin, _ValArray, _Tp >, _Tp > **asin** (const valarray< _Tp > &__v)
- constexpr float **asin** (float __x)
- constexpr long double **asin** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::asin(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **asin** (simd< _Tp, _Abi > __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **asinh** (_Tp __x)
- template<typename _Tp>
  std::complex< _Tp > **asinh** (const std::complex< _Tp > &)
- constexpr float **asinh** (float __x)
- constexpr long double **asinh** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::asinh(declval<double>())), ↩ _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **asinh** (simd< _Tp, _Abi > __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type assoc_laguerre (unsigned int __n, unsigned int __m, _Tp __x)
- float assoc_laguerref (unsigned int __n, unsigned int __m, float __x)
- long double assoc_laguerrel (unsigned int __n, unsigned int __m, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type assoc_legendre (unsigned int __l, unsigned int __m, _Tp __x)
- float assoc_legendref (unsigned int __l, unsigned int __m, float __x)
- long double assoc_legendrel (unsigned int __l, unsigned int __m, long double __x)
- template<size_t _Align, class _Tp>
  constexpr _Tp * assume_aligned (_Tp *__ptr) noexcept
- template<typename _Fn, typename... _Args>
  future< __async_result_of< _Fn, _Args... > > async (_Fn &&__fn, _Args &&... __args)
- template<typename _Fn, typename... _Args>
  future< __async_result_of< _Fn, _Args... > > async (launch __policy, _Fn &&__fn, _Args &&... __args)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **atan** (_Tp __x)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Atan, _Expr, _Dom >, typename _Dom::value_type > **atan** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  std::complex< _Tp > **atan** (const std::complex< _Tp > &)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Atan, _ValArray, _Tp >, _Tp > **atan** (const valarray< _Tp > &__v)
- constexpr float **atan** (float __x)
- constexpr long double **atan** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::atan(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **atan** (simd< _Tp, _Abi > __x)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type **atan2** (_Tp __y, _Up __x)

- template<class _Dom>
  _Expr< _BinClos< struct std::_Atan2, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _↩ Dom::value_type > **atan2** (const _Expr< _Dom, typename _Dom::value_type > &__e, const typename _Dom↩ ::value_type &__t)
- template<class _Dom>
  _Expr< _BinClos< struct std::_Atan2, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename ↩ _Dom::value_type > **atan2** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< type- name _Dom::value_type > &__v)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::_Atan2, _Expr, _Expr, _Dom1, _Dom2 >, typename _Dom1::value_type > **atan2** (const _Expr< _Dom1, typename _Dom1::value_type > &__e1, const _Expr< _Dom2, typename _Dom2↩ ::value_type > &__e2)
- template<typename _Tp, typename _Abi, typename..., typename _Arg2 = _Extra_argument_type<_Tp, _Tp, _Abi>, typename _R = _↩ Math_return_type_t< decltype(std::atan2(declval<double>(), _Arg2::declval())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **atan2** (const simd< _Tp, _Abi > &__x, const typename _Arg2::type &__y)
- template<class _Dom>
  _Expr< _BinClos< struct std::_Atan2, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename ↩ _Dom::value_type > **atan2** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom↩ ::value_type > &__e)
- template<typename _Tp>
  _Expr< _BinClos< struct std::_Atan2, _Constant, _ValArray, _Tp, _Tp >, _Tp > **atan2** (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)
- template<typename _Tp>
  _Expr< _BinClos< struct std::_Atan2, _ValArray, _Constant, _Tp, _Tp >, _Tp > **atan2** (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)
- template<typename _Tp>
  _Expr< _BinClos< struct std::_Atan2, _ValArray, _ValArray, _Tp, _Tp >, _Tp > **atan2** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::_Atan2, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _↩ Dom::value_type > **atan2** (const valarray< typename _Dom::valarray > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- constexpr float **atan2** (float __y, float __x)
- constexpr long double **atan2** (long double __y, long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **atanh** (_Tp __x)
- template<typename _Tp>
  std::complex< _Tp > **atanh** (const std::complex< _Tp > &)
- constexpr float **atanh** (float __x)
- constexpr long double **atanh** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::atanh(declval<double>())), ↩ _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **atanh** (simd< _Tp, _Abi > __x)
- template<typename _ITp>
  bool **atomic_compare_exchange_strong** (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_↩ val_t< _ITp > __i2) noexcept
- template<typename _ITp>
  bool **atomic_compare_exchange_strong** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __↩ atomic_val_t< _ITp > __i2) noexcept
- template<typename _ITp>
  bool **atomic_compare_exchange_strong_explicit** (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, _↩ _atomic_val_t< _ITp > __i2, memory_order __m1, memory_order __m2) noexcept

- template<typename _ITp>
  bool **atomic_compare_exchange_strong_explicit** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > *↩
  __i1, __atomic_val_t< _ITp > __i2, memory_order __m1, memory_order __m2) noexcept
- template<typename _ITp>
  bool **atomic_compare_exchange_weak** (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_↩
  val_t< _ITp > __i2) noexcept
- template<typename _ITp>
  bool **atomic_compare_exchange_weak** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __↩
  atomic_val_t< _ITp > __i2) noexcept
- template<typename _ITp>
  bool **atomic_compare_exchange_weak_explicit** (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __↩
  atomic_val_t< _ITp > __i2, memory_order __m1, memory_order __m2) noexcept
- template<typename _ITp>
  bool **atomic_compare_exchange_weak_explicit** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > *↩
  __i1, __atomic_val_t< _ITp > __i2, memory_order __m1, memory_order __m2) noexcept
- template<typename _ITp>
  _ITp **atomic_exchange** (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **atomic_exchange** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **atomic_exchange_explicit** (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m)
  noexcept
- template<typename _ITp>
  _ITp **atomic_exchange_explicit** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order
  __m) noexcept
- template<typename _ITp>
  _ITp **atomic_fetch_add** (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **atomic_fetch_add** (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **atomic_fetch_add_explicit** (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order __m)
  noexcept
- template<typename _ITp>
  _ITp **atomic_fetch_add_explicit** (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order
  __m) noexcept
- template<typename _ITp>
  _ITp **atomic_fetch_and** (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **atomic_fetch_and** (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **atomic_fetch_and_explicit** (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order
  __m) noexcept
- template<typename _ITp>
  _ITp **atomic_fetch_and_explicit** (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __↩
  i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **atomic_fetch_or** (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **atomic_fetch_or** (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **atomic_fetch_or_explicit** (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order
  __m) noexcept

- template<typename _ITp>
  _ITp **atomic_fetch_or_explicit** (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **atomic_fetch_sub** (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **atomic_fetch_sub** (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **atomic_fetch_sub_explicit** (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **atomic_fetch_sub_explicit** (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **atomic_fetch_xor** (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **atomic_fetch_xor** (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **atomic_fetch_xor_explicit** (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **atomic_fetch_xor_explicit** (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- void **atomic_flag_clear** (atomic_flag *__a) noexcept
- void **atomic_flag_clear** (volatile atomic_flag *__a) noexcept
- void **atomic_flag_clear_explicit** (atomic_flag *__a, memory_order __m) noexcept
- void **atomic_flag_clear_explicit** (volatile atomic_flag *__a, memory_order __m) noexcept
- bool **atomic_flag_test_and_set** (atomic_flag *__a) noexcept
- bool **atomic_flag_test_and_set** (volatile atomic_flag *__a) noexcept
- bool **atomic_flag_test_and_set_explicit** (atomic_flag *__a, memory_order __m) noexcept
- bool **atomic_flag_test_and_set_explicit** (volatile atomic_flag *__a, memory_order __m) noexcept
- template<typename _ITp>
  void **atomic_init** (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  void **atomic_init** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  bool **atomic_is_lock_free** (const atomic< _ITp > *__a) noexcept
- template<typename _ITp>
  bool **atomic_is_lock_free** (const volatile atomic< _ITp > *__a) noexcept
- template<typename _ITp>
  _ITp **atomic_load** (const atomic< _ITp > *__a) noexcept
- template<typename _ITp>
  _ITp **atomic_load** (const volatile atomic< _ITp > *__a) noexcept
- template<typename _ITp>
  _ITp **atomic_load_explicit** (const atomic< _ITp > *__a, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **atomic_load_explicit** (const volatile atomic< _ITp > *__a, memory_order __m) noexcept
- void **atomic_signal_fence** (memory_order __m) noexcept
- template<typename _ITp>
  void **atomic_store** (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  void **atomic_store** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept

- template<typename _ITp>
  void **atomic_store_explicit** (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  void **atomic_store_explicit** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- void **atomic_thread_fence** (memory_order __m) noexcept
- template<typename _Container>
  constexpr back_insert_iterator< _Container > back_inserter (_Container &__x)
- template<typename _ForwardIterator>
  basic_regex (_ForwardIterator, _ForwardIterator, regex_constants::syntax_option_type={}) -> basic_regex< typename iterator_traits< _ForwardIterator >::value_type >
- template<typename _InputIterator, typename _CharT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_CharT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  basic_string (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> basic_string< _CharT, char_traits< _CharT >, _Allocator >
- template<typename _CharT, typename _Traits, typename _Allocator = allocator<_CharT>, typename = _RequireAllocator<_Allocator>>
  basic_string (basic_string_view< _CharT, _Traits >, const _Allocator &=_Allocator()) -> basic_string< _CharT, _Traits, _Allocator >
- template<typename _CharT, typename _Traits, typename _Allocator = allocator<_CharT>, typename = _RequireAllocator<_Allocator>>
  basic_string (basic_string_view< _CharT, _Traits >, typename basic_string< _CharT, _Traits, _Allocator >::size_type, typename basic_string< _CharT, _Traits, _Allocator >::size_type, const _Allocator &=_Allocator()) -> basic_string< _CharT, _Traits, _Allocator >
- template<typename _Operation>
  void **basic_string< _CharT, _Traits, _Alloc >__resize_and_overwrite** (const size_type __n, _Operation __op)
- template<typename _Container>
  constexpr auto begin (_Container &__cont) noexcept(noexcept(__cont.begin())) -> decltype(__cont.begin())
- template<typename _Tp, size_t _Nm>
  constexpr _Tp * begin (_Tp(&__arr)[_Nm]) noexcept
- template<typename _Container>
  constexpr auto begin (const _Container &__cont) noexcept(noexcept(__cont.begin())) -> decltype(__cont.begin())
- template<class _Tp>
  const _Tp * begin (const valarray< _Tp > &__va) noexcept
- template<class _Tp>
  _Tp * begin (valarray< _Tp > &__va) noexcept
- template<class _Tp>
  constexpr const _Tp * begin (initializer_list< _Tp > __ils) noexcept
- template<typename _Tpa, typename _Tpb>
  __gnu_cxx::__promote_2< _Tpa, _Tpb >::__type beta (_Tpa __a, _Tpb __b)
- float betaf (float __a, float __b)
- long double betal (long double __a, long double __b)
- template<typename _FIter, typename _Tp>
  constexpr bool **binary_search** (_FIter, _FIter, const _Tp &)
- template<typename _FIter, typename _Tp, typename _Compare>
  constexpr bool **binary_search** (_FIter, _FIter, const _Tp &, _Compare)
- template<typename _ForwardIterator, typename _Tp>
  constexpr bool binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)
- template<typename _ForwardIterator, typename _Tp, typename _Compare>
  constexpr bool binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)
- template<typename _Func, typename... _BoundArgs>
  constexpr _Bind_helper< __is_socketlike< _Func >::value, _Func, _BoundArgs... >::type bind (_Func &&__f, _BoundArgs &&... __args)

- template<typename _Result, typename _Func, typename... _BoundArgs>
  constexpr _Bindres_helper< _Result, _Func, _BoundArgs... >::type bind (_Func &&__f, _BoundArgs &&... __↩
  args)
- template<typename _Operation, typename _Tp>
  binder1st< _Operation > bind1st (const _Operation &__fn, const _Tp &__x)
- template<typename _Operation, typename _Tp>
  binder2nd< _Operation > bind2nd (const _Operation &__fn, const _Tp &__x)
- ios_base & boolalpha (ios_base &__base)
- template<typename _Callable, typename... _Args>
  void call_once (once_flag &__once, _Callable &&__f, _Args &&... __args)
- template<typename _Container>
  constexpr auto cbegin (const _Container &__cont) noexcept(noexcept(std::begin(__cont))) -> decltype(std::begin(↩
  __cont))
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **cbrt** (_Tp __x)
- constexpr float **cbrt** (float __x)
- constexpr long double **cbrt** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::cbrt(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **cbrt** (simd< _Tp, _Abi > __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **ceil** (_Tp __x)
- constexpr float **ceil** (float __x)
- constexpr long double **ceil** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::ceil(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **ceil** (simd< _Tp, _Abi > __x)
- template<typename _Container>
  constexpr auto cend (const _Container &__cont) noexcept(noexcept(std::end(__cont))) -> decltype(std::end(_↩
  _cont))
- template<typename _Tp>
  constexpr const _Tp & clamp (const _Tp &__val, const _Tp &__lo, const _Tp &__hi)
- template<typename _Tp, typename _Compare>
  constexpr const _Tp & clamp (const _Tp &__val, const _Tp &__lo, const _Tp &__hi, _Compare __comp)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type comp_ellint_1 (_Tp __k)
- float comp_ellint_1f (float __k)
- long double comp_ellint_1l (long double __k)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type comp_ellint_2 (_Tp __k)
- float comp_ellint_2f (float __k)
- long double comp_ellint_2l (long double __k)
- template<typename _Tp, typename _Tpn>
  __gnu_cxx::__promote_2< _Tp, _Tpn >::__type comp_ellint_3 (_Tp __k, _Tpn __nu)
- float comp_ellint_3f (float __k, float __nu)
- long double comp_ellint_3l (long double __k, long double __nu)
- template<typename _Tp>
  constexpr std::complex< typename __gnu_cxx::__promote< _Tp >::__type > **conj** (_Tp __x)
- template<typename _Tp>
  constexpr complex< _Tp > conj (const complex< _Tp > &)

- template<typename _Tp, typename _Tp1, _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > const_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept
- template<typename _Tp, typename... _Args>
  requires (!is_unbounded_array_v<_Tp>) && requires { ::new((void*)0) _Tp(std::declval<_Args>()...); }
  constexpr _Tp * **construct_at** (_Tp *__location, _Args &&... __args) noexcept(noexcept(::new((void *) 0) _Tp(std::declval< _Args >()...)))
- template<typename _II, typename _OI>
  constexpr _OI copy (_II __first, _II __last, _OI __result)
- template<typename _IIter, typename _OIter>
  constexpr _OIter **copy** (_IIter, _IIter, _OIter)
- template<typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type **copy** (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, ostreambuf_iterator< _CharT > __result)
- template<typename _BI1, typename _BI2>
  constexpr _BI2 copy_backward (_BI1 __first, _BI1 __last, _BI2 __result)
- template<typename _BIter1, typename _BIter2>
  constexpr _BIter2 **copy_backward** (_BIter1, _BIter1, _BIter2)
- template<typename _IIter, typename _OIter, typename _Predicate>
  constexpr _OIter **copy_if** (_IIter, _IIter, _OIter, _Predicate)
- template<typename _InputIterator, typename _OutputIterator, typename _Predicate>
  constexpr _OutputIterator copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)
- template<typename _IIter, typename _Size, typename _OIter>
  constexpr _OIter **copy_n** (_IIter, _Size, _OIter)
- template<typename _InputIterator, typename _Size, typename _OutputIterator>
  constexpr _OutputIterator copy_n (_InputIterator __first, _Size __n, _OutputIterator __result)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type **copysign** (_Tp __x, _Up __y)
- template<typename _Tp, typename _Abi, typename = __detail::__odr_helper>
  enable_if_t< is_floating_point_v< _Tp >, simd< _Tp, _Abi > > **copysign** (const simd< _Tp, _Abi > &__x, const simd< _Tp, _Abi > &__y)
- constexpr float **copysign** (float __x, float __y)
- constexpr long double **copysign** (long double __x, long double __y)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **cos** (_Tp __x)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Cos, _Expr, _Dom >, typename _Dom::value_type > **cos** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  complex< _Tp > cos (const complex< _Tp > &)
- template<typename _Tp, typename _Abi, typename = __detail::__odr_helper>
  enable_if_t< is_floating_point_v< _Tp >, simd< _Tp, _Abi > > **cos** (const simd< _Tp, _Abi > &__x)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Cos, _ValArray, _Tp >, _Tp > **cos** (const valarray< _Tp > &__v)
- constexpr float **cos** (float __x)
- constexpr long double **cos** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **cosh** (_Tp __x)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Cosh, _Expr, _Dom >, typename _Dom::value_type > **cosh** (const _Expr< _Dom, typename _Dom::value_type > &__e)

- template<typename _Tp>
  complex< _Tp > cosh (const complex< _Tp > &)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Cosh, _ValArray, _Tp >, _Tp > **cosh** (const valarray< _Tp > &__v)
- constexpr float **cosh** (float __x)
- constexpr long double **cosh** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::cosh(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **cosh** (simd< _Tp, _Abi > __x)
- template<typename _IIter, typename _Tp>
  constexpr iterator_traits< _IIter >::difference_type **count** (_IIter, _IIter, const _Tp &)
- template<typename _InputIterator, typename _Tp>
  constexpr iterator_traits< _InputIterator >::difference_type count (_InputIterator __first, _InputIterator __last, const _Tp &__value)
- template<typename _IIter, typename _Predicate>
  constexpr iterator_traits< _IIter >::difference_type **count_if** (_IIter, _IIter, _Predicate)
- template<typename _InputIterator, typename _Predicate>
  constexpr iterator_traits< _InputIterator >::difference_type count_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _Container>
  constexpr auto crbegin (const _Container &__cont) noexcept(noexcept(std::rbegin(__cont))) -> decltype(std::rbegin(↩ __cont))
- template<typename _Container>
  constexpr auto crend (const _Container &__cont) noexcept(noexcept(std::rend(__cont))) -> decltype(std::rend(↩ __cont))
- exception_ptr current_exception () noexcept
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl_bessel_i (_Tpnu __nu, _Tp __x)
- float cyl_bessel_if (float __nu, float __x)
- long double cyl_bessel_il (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl_bessel_j (_Tpnu __nu, _Tp __x)
- float cyl_bessel_jf (float __nu, float __x)
- long double cyl_bessel_jl (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl_bessel_k (_Tpnu __nu, _Tp __x)
- float cyl_bessel_kf (float __nu, float __x)
- long double cyl_bessel_kl (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl_neumann (_Tpnu __nu, _Tp __x)
- float cyl_neumannf (float __nu, float __x)
- long double cyl_neumannl (long double __nu, long double __x)
- template<typename _Container>
  constexpr auto data (_Container &__cont) noexcept(noexcept(__cont.data())) -> decltype(__cont.data())
- template<typename _Tp, size_t _Nm>
  constexpr _Tp * data (_Tp(&__array)[_Nm]) noexcept
- template<typename _Container>
  constexpr auto data (const _Container &__cont) noexcept(noexcept(__cont.data())) -> decltype(__cont.data())
- template<typename _Tp>
  constexpr const _Tp * data (initializer_list< _Tp > __il) noexcept
- ios_base & dec (ios_base &__base)

- void declare_no_pointers (char *, size_t)
- void declare_reachable (void *)
- template<typename _Tp>
  auto declval () noexcept -> decltype(__declval< _Tp >(0))
- ios_base & defaultfloat (ios_base &__base)
- template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  deque (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> deque< _ValT, _Allocator >
- template<typename _ForwardIterator>
  constexpr void **destroy** (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _Tp>
  constexpr void **destroy_at** (_Tp *__location)
- template<typename _ForwardIterator, typename _Size>
  constexpr _ForwardIterator **destroy_n** (_ForwardIterator __first, _Size __count)
- template<typename _InputIterator>
  constexpr iterator_traits< _InputIterator >::difference_type distance (_InputIterator __first, _InputIterator __last)
- ldiv_t **div** (long __i, long __j) noexcept
- template<typename _Tp, typename _Tp1, _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > dynamic_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept
- template<typename _Tp, typename _Tpp>
  __gnu_cxx::__promote_2< _Tp, _Tpp >::__type ellint_1 (_Tp __k, _Tpp __phi)
- float ellint_1f (float __k, float __phi)
- long double ellint_1l (long double __k, long double __phi)
- template<typename _Tp, typename _Tpp>
  __gnu_cxx::__promote_2< _Tp, _Tpp >::__type ellint_2 (_Tp __k, _Tpp __phi)
- float ellint_2f (float __k, float __phi)
- long double ellint_2l (long double __k, long double __phi)
- template<typename _Tp, typename _Tpn, typename _Tpp>
  __gnu_cxx::__promote_3< _Tp, _Tpn, _Tpp >::__type ellint_3 (_Tp __k, _Tpn __nu, _Tpp __phi)
- float ellint_3f (float __k, float __nu, float __phi)
- long double ellint_3l (long double __k, long double __nu, long double __phi)
- template<typename _Container>
  constexpr auto empty (const _Container &__cont) noexcept(noexcept(__cont.empty())) -> decltype(__cont.↩ empty())
- template<typename _Tp, size_t _Nm>
  constexpr bool empty (const _Tp(&)[_Nm]) noexcept
- template<typename _Tp>
  constexpr bool empty (initializer_list< _Tp > __il) noexcept
- template<typename _Container>
  constexpr auto end (_Container &__cont) noexcept(noexcept(__cont.end())) -> decltype(__cont.end())
- template<typename _Tp, size_t _Nm>
  constexpr _Tp * end (_Tp(&__arr)[_Nm]) noexcept
- template<typename _Container>
  constexpr auto end (const _Container &__cont) noexcept(noexcept(__cont.end())) -> decltype(__cont.end())
- template<class _Tp>
  const _Tp * end (const valarray< _Tp > &__va) noexcept
- template<class _Tp>
  _Tp * end (valarray< _Tp > &__va) noexcept
- template<class _Tp>
  constexpr const _Tp * end (initializer_list< _Tp > __ils) noexcept
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & endl (basic_ostream< _CharT, _Traits > &__os)

- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & ends (basic_ostream< _CharT, _Traits > &__os)
- template<typename _II1, typename _II2>
  constexpr bool equal (_II1 __first1, _II1 __last1, _II2 __first2)
- template<typename _II1, typename _II2>
  constexpr bool equal (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)
- template<typename _IIter1, typename _IIter2, typename _BinaryPredicate>
  constexpr bool equal (_IIter1 __first1, _IIter1 __last1, _IIter2 __first2, _BinaryPredicate __binary_pred)
- template<typename _IIter1, typename _IIter2, typename _BinaryPredicate>
  constexpr bool equal (_IIter1 __first1, _IIter1 __last1, _IIter2 __first2, _IIter2 __last2, _BinaryPredicate __binary↩
  _pred)
- template<typename _IIter1, typename _IIter2>
  constexpr bool **equal** (_IIter1, _IIter1, _IIter2)
- template<typename _FIter, typename _Tp>
  constexpr pair< _FIter, _FIter > **equal_range** (_FIter, _FIter, const _Tp &)
- template<typename _FIter, typename _Tp, typename _Compare>
  constexpr pair< _FIter, _FIter > **equal_range** (_FIter, _FIter, const _Tp &, _Compare)
- template<typename _ForwardIterator, typename _Tp>
  constexpr pair< _ForwardIterator, _ForwardIterator > equal_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)
- template<typename _ForwardIterator, typename _Tp, typename _Compare>
  constexpr pair< _ForwardIterator, _ForwardIterator > equal_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc, typename _Predicate>
  map< _Key, _Tp, _Compare, _Alloc >::size_type **erase_if** (map< _Key, _Tp, _Compare, _Alloc > &__cont, _Predicate __pred)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc, typename _Predicate>
  multimap< _Key, _Tp, _Compare, _Alloc >::size_type **erase_if** (multimap< _Key, _Tp, _Compare, _Alloc > &__cont, _Predicate __pred)
- template<typename _Key, typename _Compare, typename _Alloc, typename _Predicate>
  multiset< _Key, _Compare, _Alloc >::size_type **erase_if** (multiset< _Key, _Compare, _Alloc > &__cont, _↩
  Predicate __pred)
- template<typename _Key, typename _Compare, typename _Alloc, typename _Predicate>
  set< _Key, _Compare, _Alloc >::size_type **erase_if** (set< _Key, _Compare, _Alloc > &__cont, _Predicate __↩
  pred)
- template<typename _Key, typename _Tp, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>
  unordered_map< _Key, _Tp, _Hash, _CPred, _Alloc >::size_type **erase_if** (unordered_map< _Key, _Tp, _Hash, _CPred, _Alloc > &__cont, _Predicate __pred)
- template<typename _Key, typename _Tp, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>
  unordered_multimap< _Key, _Tp, _Hash, _CPred, _Alloc >::size_type **erase_if** (unordered_multimap< _Key, _Tp, _Hash, _CPred, _Alloc > &__cont, _Predicate __pred)
- template<typename _Key, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>
  unordered_multiset< _Key, _Hash, _CPred, _Alloc >::size_type **erase_if** (unordered_multiset< _Key, _Hash, _CPred, _Alloc > &__cont, _Predicate __pred)
- template<typename _Key, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>
  unordered_set< _Key, _Hash, _CPred, _Alloc >::size_type **erase_if** (unordered_set< _Key, _Hash, _CPred, _Alloc > &__cont, _Predicate __pred)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **erf** (_Tp __x)
- constexpr float **erf** (float __x)
- constexpr long double **erf** (long double __x)

- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::erf(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **erf** (simd< _Tp, _Abi > __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **erfc** (_Tp __x)
- constexpr float **erfc** (float __x)
- constexpr long double **erfc** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::erfc(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **erfc** (simd< _Tp, _Abi > __x)
- template<typename _InputIterator, typename _OutputIterator, typename _Tp>
  constexpr _OutputIterator exclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Tp __init)
- template<typename _InputIterator, typename _OutputIterator, typename _Tp, typename _BinaryOperation>
  constexpr _OutputIterator exclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Tp __init, _BinaryOperation __binary_op)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **exp** (_Tp __x)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Exp, _Expr, _Dom >, typename _Dom::value_type > **exp** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  complex< _Tp > exp (const complex< _Tp > &)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Exp, _ValArray, _Tp >, _Tp > **exp** (const valarray< _Tp > &__v)
- constexpr float **exp** (float __x)
- constexpr long double **exp** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::exp(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **exp** (simd< _Tp, _Abi > __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **exp2** (_Tp __x)
- constexpr float **exp2** (float __x)
- constexpr long double **exp2** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::exp2(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **exp2** (simd< _Tp, _Abi > __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type expint (_Tp __x)
- float expintf (float __x)
- long double expintl (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **expm1** (_Tp __x)
- constexpr float **expm1** (float __x)
- constexpr long double **expm1** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::expm1(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **expm1** (simd< _Tp, _Abi > __x)

- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **fabs** (_Tp __x)
- template<typename _Tp>
  _Tp fabs (const std::complex< _Tp > &__z)
- constexpr float **fabs** (float __x)
- constexpr long double **fabs** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::fabs(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **fabs** (simd< _Tp, _Abi > __x)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type **fdim** (_Tp __x, _Up __y)
- template<typename _Tp, typename _Abi, typename..., typename _Arg2 = _Extra_argument_type<_Tp, _Tp, _Abi>, typename _R = _↩
  Math_return_type_t< decltype(std::fdim(declval<double>(), _Arg2::declval())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **fdim** (const simd< _Tp, _Abi > &__x, const typename _Arg2::type &__y)
- constexpr float **fdim** (float __x, float __y)
- constexpr long double **fdim** (long double __x, long double __y)
- template<typename _FIter, typename _Tp>
  constexpr void **fill** (_FIter, _FIter, const _Tp &)
- template<typename _ForwardIterator, typename _Tp>
  constexpr void fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)
- template<typename _OI, typename _Size, typename _Tp>
  constexpr _OI fill_n (_OI __first, _Size __n, const _Tp &__value)
- template<typename _OIter, typename _Size, typename _Tp>
  constexpr _OIter **fill_n** (_OIter, _Size, const _Tp &)
- template<typename _IIter, typename _Tp>
  constexpr _IIter **find** (_IIter, _IIter, const _Tp &)
- template<typename _InputIterator, typename _Tp>
  constexpr _InputIterator find (_InputIterator __first, _InputIterator __last, const _Tp &__val)
- template<typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, istreambuf_iterator< _CharT > >::__type **find** (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, const _CharT &__val)
- template<typename _FIter1, typename _FIter2>
  constexpr _FIter1 **find_end** (_FIter1, _FIter1, _FIter2, _FIter2)
- template<typename _FIter1, typename _FIter2, typename _BinaryPredicate>
  constexpr _FIter1 **find_end** (_FIter1, _FIter1, _FIter2, _FIter2, _BinaryPredicate)
- template<typename _ForwardIterator1, typename _ForwardIterator2>
  constexpr _ForwardIterator1 find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)
- template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>
  constexpr _ForwardIterator1 find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __comp)
- template<typename _FIter1, typename _FIter2>
  constexpr _FIter1 **find_first_of** (_FIter1, _FIter1, _FIter2, _FIter2)
- template<typename _FIter1, typename _FIter2, typename _BinaryPredicate>
  constexpr _FIter1 **find_first_of** (_FIter1, _FIter1, _FIter2, _FIter2, _BinaryPredicate)
- template<typename _InputIterator, typename _ForwardIterator>
  constexpr _InputIterator find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _↩
  ForwardIterator __last2)
- template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate>
  constexpr _InputIterator find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _↩
  ForwardIterator __last2, _BinaryPredicate __comp)

- template<typename _IIter, typename _Predicate>
  constexpr _IIter **find_if** (_IIter, _IIter, _Predicate)
- template<typename _InputIterator, typename _Predicate>
  constexpr _InputIterator find_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _IIter, typename _Predicate>
  constexpr _IIter **find_if_not** (_IIter, _IIter, _Predicate)
- template<typename _InputIterator, typename _Predicate>
  constexpr _InputIterator find_if_not (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- ios_base & fixed (ios_base &__base)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **floor** (_Tp __x)
- constexpr float **floor** (float __x)
- constexpr long double **floor** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::floor(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **floor** (simd< _Tp, _Abi > __x)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & flush (basic_ostream< _CharT, _Traits > &__os)
- template<typename _Tp, typename _Up, typename _Vp>
  constexpr __gnu_cxx::__promote_3< _Tp, _Up, _Vp >::__type **fma** (_Tp __x, _Up __y, _Vp __z)
- template<typename _Tp, typename _Abi, typename..., typename _Arg2 = _Extra_argument_type<_Tp, _Tp, _Abi>, typename _Arg3 = _Extra_argument_type<_Tp, _Tp, _Abi>, typename _R = _Math_return_type_t< decltype(std::fma(declval<double>(), _Arg2::declval(), _Arg3::declval())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **fma** (const simd< _Tp, _Abi > &__x, const typename _Arg2::type &__y, const typename _Arg3::type &__z)
- constexpr float **fma** (float __x, float __y, float __z)
- constexpr long double **fma** (long double __x, long double __y, long double __z)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type **fmax** (_Tp __x, _Up __y)
- template<typename _Tp, typename _Abi, typename..., typename _Arg2 = _Extra_argument_type<_Tp, _Tp, _Abi>, typename _R = _↩ Math_return_type_t< decltype(std::fmax(declval<double>(), _Arg2::declval())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **fmax** (const simd< _Tp, _Abi > &__x, const typename _Arg2::type &__y)
- constexpr float **fmax** (float __x, float __y)
- constexpr long double **fmax** (long double __x, long double __y)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type **fmin** (_Tp __x, _Up __y)
- template<typename _Tp, typename _Abi, typename..., typename _Arg2 = _Extra_argument_type<_Tp, _Tp, _Abi>, typename _R = _↩ Math_return_type_t< decltype(std::fmin(declval<double>(), _Arg2::declval())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **fmin** (const simd< _Tp, _Abi > &__x, const typename _Arg2::type &__y)
- constexpr float **fmin** (float __x, float __y)
- constexpr long double **fmin** (long double __x, long double __y)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type **fmod** (_Tp __x, _Up __y)
- template<typename _Tp, typename _Abi, typename..., typename _Arg2 = _Extra_argument_type<_Tp, _Tp, _Abi>, typename _R = _↩ Math_return_type_t< decltype(std::fmod(declval<double>(), _Arg2::declval())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **fmod** (const simd< _Tp, _Abi > &__x, const typename _Arg2::type &__y)
- constexpr float **fmod** (float __x, float __y)
- constexpr long double **fmod** (long double __x, long double __y)

- template<typename _IIter, typename _Funct>
  constexpr _Funct **for_each** (_IIter, _IIter, _Funct)
- template<typename _InputIterator, typename _Function>
  constexpr _Function for_each (_InputIterator __first, _InputIterator __last, _Function __f)
- template<typename _InputIterator, typename _Size, typename _Function>
  constexpr _InputIterator for_each_n (_InputIterator __first, _Size __n, _Function __f)
- template<typename _Tp>
  constexpr _Tp && forward (typename std::remove_reference< _Tp >::type &&__t) noexcept
- template<typename _Tp>
  constexpr _Tp && forward (typename std::remove_reference< _Tp >::type &__t) noexcept
- template<typename... _Elements>
  constexpr tuple< _Elements &&... > forward_as_tuple (_Elements &&... __args) noexcept
- template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  forward_list (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> forward_list< _ValT, _Allocator >
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, int >::__type **fpclassify** (_Tp __x)
- constexpr int **fpclassify** (double __x)
- constexpr int **fpclassify** (float __x)
- constexpr int **fpclassify** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **frexp** (_Tp __x, int *__exp)
- template<typename _Tp, typename _Abi, typename = __detail::__odr_helper>
  enable_if_t< is_floating_point_v< _Tp >, simd< _Tp, _Abi > > frexp (const simd< _Tp, _Abi > &__x, Samesize< int, simd< _Tp, _Abi > > *__exp)
- float **frexp** (float __x, int *__exp)
- long double **frexp** (long double __x, int *__exp)
- template<typename _Tp, enable_if_t< __or_< __is_standard_integer< _Tp >, is_same< char, remove_cv_t< _Tp > > >::value, int > = 0>
  constexpr from_chars_result from_chars (const char *__first, const char *__last, _Tp &__value, int __base=10)
- from_chars_result **from_chars** (const char *__first, const char *__last, double &__value, chars_format __fmt=chars_format::general) noexcept
- from_chars_result **from_chars** (const char *__first, const char *__last, float &__value, chars_format __fmt=chars_format::general) noexcept
- from_chars_result **from_chars** (const char *__first, const char *__last, long double &__value, chars_format __fmt=chars_format::general) noexcept
- template<typename _Container>
  constexpr front_insert_iterator< _Container > front_inserter (_Container &__x)
- template<typename _Fn, typename _Signature = __function_guide_t<_Fn, decltype(&_Fn::operator())>>
  function (_Fn) -> function< _Signature >
- template<typename _Res, typename... _ArgTypes>
  function (_Res(*)(_ArgTypes...)) -> function< _Res(_ArgTypes...)>
- const error_category & future_category () noexcept
- template<typename _FIter, typename _Generator>
  constexpr void **generate** (_FIter, _FIter, _Generator)
- template<typename _ForwardIterator, typename _Generator>
  constexpr void generate (_ForwardIterator __first, _ForwardIterator __last, _Generator __gen)
- template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator>
  _RealType generate_canonical (_UniformRandomNumberGenerator &__g)
- template<typename _OIter, typename _Size, typename _Generator>
  constexpr _OIter **generate_n** (_OIter, _Size, _Generator)

- template<typename _OutputIterator, typename _Size, typename _Generator>
  constexpr _OutputIterator generate_n (_OutputIterator __first, _Size __n, _Generator __gen)
- const error_category & generic_category () noexcept
- template<std::size_t _Int, typename _Tp, std::size_t _Nm>
  constexpr _Tp && **get** (array< _Tp, _Nm > &&__arr) noexcept
- template<std::size_t _Int, typename _Tp, std::size_t _Nm>
  constexpr _Tp & **get** (array< _Tp, _Nm > &__arr) noexcept
- template<std::size_t _Int, typename _Tp, std::size_t _Nm>
  constexpr const _Tp && **get** (const array< _Tp, _Nm > &&__arr) noexcept
- template<std::size_t _Int, typename _Tp, std::size_t _Nm>
  constexpr const _Tp & **get** (const array< _Tp, _Nm > &__arr) noexcept
- template<size_t _Num, class _It, class _Sent, subrange_kind _Kind>
  requires ((_Num == 0 && copyable<_It>) || _Num == 1)
  constexpr auto **get** (const subrange< _It, _Sent, _Kind > &__r)
- template<size_t __i, typename... _Elements>
  constexpr const __tuple_element_t< __i, tuple< _Elements... > > && get (const tuple< _Elements... > &&__t) noexcept
- template<size_t __i, typename... _Elements>
  constexpr const __tuple_element_t< __i, tuple< _Elements... > > & get (const tuple< _Elements... > &__t) noexcept
- template<size_t __i, typename... _Elements>
  constexpr __tuple_element_t< __i, tuple< _Elements... > > && get (tuple< _Elements... > &&__t) noexcept
- template<size_t __i, typename... _Elements>
  constexpr __tuple_element_t< __i, tuple< _Elements... > > & get (tuple< _Elements... > &__t) noexcept
- Catalogs & **get_catalogs** ()
- template<typename _Del, typename _Tp, _Lock_policy _Lp>
  _Del * **get_deleter** (const __shared_ptr< _Tp, _Lp > &__p) noexcept
- template<typename _Del, typename _Tp>
  _Del * get_deleter (const shared_ptr< _Tp > &__p) noexcept
- template<typename _MoneyT>
  _Get_money< _MoneyT > get_money (_MoneyT &__mon, bool __intl=false)
- new_handler get_new_handler () noexcept
- pointer_safety get_pointer_safety () noexcept
- template<typename _Tp>
  pair< _Tp *, ptrdiff_t > get_temporary_buffer (ptrdiff_t __len) noexcept
- terminate_handler get_terminate () noexcept
- template<typename _CharT>
  _Get_time< _CharT > get_time (std::tm *__tmb, const _CharT *__fmt)
- unexpected_handler get_unexpected () noexcept
- template<typename _CharT, typename _Traits, typename _Alloc>
  basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > &&__is, basic_string< _CharT, _Traits, _Alloc > &__str)
- template<typename _CharT, typename _Traits, typename _Alloc>
  basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > &&__is, basic_string< _CharT, _Traits, _Alloc > &__str, _CharT __delim)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str, _CharT __delim)

- template<typename _CharT, typename _Traits, typename _Alloc>
  basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str)
- template<typename _CharT, typename _Traits, typename _Alloc>
  basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str, _CharT __delim)
- template<> basic_istream< char > & **getline** (basic_istream< char > &__in, basic_string< char > &__str, char __delim)
- template<> basic_istream< wchar_t > & **getline** (basic_istream< wchar_t > &__in, basic_string< wchar_t > &__str, wchar_t __delim)
- template<typename _Facet>
  bool has_facet (const locale &__loc)
- template bool **has_facet**< **codecvt**< **char, char, mbstate_t** > > (const locale &)
- template bool **has_facet**< **codecvt**< **wchar_t, char, mbstate_t** > > (const locale &)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type hermite (unsigned int __n, _Tp __x)
- float hermitef (unsigned int __n, float __x)
- long double hermitel (unsigned int __n, long double __x)
- ios_base & hex (ios_base &__base)
- ios_base & hexfloat (ios_base &__base)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type **hypot** (_Tp __x, _Up __y)
- template<typename _Tp, typename _Abi>
  _GLIBCXX_SIMD_INTRINSIC simd< _Tp, _Abi > **hypot** (const simd< _Tp, _Abi > &__x, const simd< _Tp, _Abi > &__y)
- constexpr float **hypot** (float __x, float __y)
- constexpr long double **hypot** (long double __x, long double __y)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, int >::__type **ilogb** (_Tp __x)
- constexpr int **ilogb** (float __x)
- constexpr int **ilogb** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::ilogb(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **ilogb** (simd< _Tp, _Abi > __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__promote< _Tp >::__type **imag** (_Tp)
- template<typename _Tp>
  constexpr _Tp **imag** (const complex< _Tp > &__z)
- template<typename _IIter1, typename _IIter2>
  constexpr bool **includes** (_IIter1, _IIter1, _IIter2, _IIter2)
- template<typename _IIter1, typename _IIter2, typename _Compare>
  constexpr bool **includes** (_IIter1, _IIter1, _IIter2, _IIter2, _Compare)
- template<typename _InputIterator1, typename _InputIterator2>
  constexpr bool includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)
- template<typename _InputIterator1, typename _InputIterator2, typename _Compare>
  constexpr bool includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp)
- template<typename _InputIterator, typename _OutputIterator>
  constexpr _OutputIterator inclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result)

- template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation>
  constexpr _OutputIterator inclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op)
- template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation, typename _Tp>
  constexpr _OutputIterator inclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op, _Tp __init)
- template<typename _InputIterator1, typename _InputIterator2, typename _Tp>
  constexpr _Tp inner_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init)
- template<typename _InputIterator1, typename _InputIterator2, typename _Tp, typename _BinaryOperation1, typename _Binary↩ Operation2>
  constexpr _Tp inner_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init, _BinaryOperation1 __binary_op1, _BinaryOperation2 __binary_op2)
- template<typename _BidirectionalIterator>
  void inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last)
- template<typename _BidirectionalIterator, typename _Compare>
  void inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, ↩ _Compare __comp)
- template<typename _BIter>
  void **inplace_merge** (_BIter, _BIter, _BIter)
- template<typename _BIter, typename _Compare>
  void **inplace_merge** (_BIter, _BIter, _BIter, _Compare)
- template<typename _Container>
  constexpr insert_iterator< _Container > inserter (_Container &__x, std::__detail::__range_iter_t< _Container > __i)
- ios_base & internal (ios_base &__base)
- const error_category & **iostream_category** () noexcept
- template<typename _ForwardIterator, typename _Tp>
  constexpr void iota (_ForwardIterator __first, _ForwardIterator __last, _Tp __value)
- constexpr bool **is_eq** (partial_ordering __cmp) noexcept
- constexpr bool **is_gt** (partial_ordering __cmp) noexcept
- constexpr bool **is_gteq** (partial_ordering __cmp) noexcept
- template<typename _RAIter>
  constexpr bool **is_heap** (_RAIter, _RAIter)
- template<typename _RAIter, typename _Compare>
  constexpr bool **is_heap** (_RAIter, _RAIter, _Compare)
- template<typename _RandomAccessIterator>
  constexpr bool is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr bool is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _RAIter>
  constexpr _RAIter **is_heap_until** (_RAIter, _RAIter)
- template<typename _RAIter, typename _Compare>
  constexpr _RAIter **is_heap_until** (_RAIter, _RAIter, _Compare)
- template<typename _RandomAccessIterator>
  constexpr _RandomAccessIterator is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __↩ last)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr _RandomAccessIterator is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __↩ last, _Compare __comp)
- constexpr bool **is_lt** (partial_ordering __cmp) noexcept
- constexpr bool **is_lteq** (partial_ordering __cmp) noexcept
- constexpr bool **is_neq** (partial_ordering __cmp) noexcept

- template<typename _IIter, typename _Predicate>
  constexpr bool **is_partitioned** (_IIter, _IIter, _Predicate)
- template<typename _InputIterator, typename _Predicate>
  constexpr bool is_partitioned (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _FIter1, typename _FIter2>
  constexpr bool **is_permutation** (_FIter1, _FIter1, _FIter2)
- template<typename _FIter1, typename _FIter2, typename _BinaryPredicate>
  constexpr bool **is_permutation** (_FIter1, _FIter1, _FIter2, _BinaryPredicate)
- template<typename _ForwardIterator1, typename _ForwardIterator2>
  constexpr bool is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)
- template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>
  constexpr bool is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _BinaryPredicate __pred)
- template<typename _ForwardIterator1, typename _ForwardIterator2>
  constexpr bool is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)
- template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>
  constexpr bool is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __pred)
- template<typename _FIter>
  constexpr bool **is_sorted** (_FIter, _FIter)
- template<typename _FIter, typename _Compare>
  constexpr bool **is_sorted** (_FIter, _FIter, _Compare)
- template<typename _ForwardIterator>
  constexpr bool is_sorted (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _Compare>
  constexpr bool is_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)
- template<typename _FIter>
  constexpr _FIter **is_sorted_until** (_FIter, _FIter)
- template<typename _FIter, typename _Compare>
  constexpr _FIter **is_sorted_until** (_FIter, _FIter, _Compare)
- template<typename _ForwardIterator>
  constexpr _ForwardIterator is_sorted_until (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _Compare>
  constexpr _ForwardIterator is_sorted_until (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)
- template<typename _CharT>
  bool isalnum (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool isalpha (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool isblank (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool iscntrl (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool isdigit (_CharT __c, const locale &__loc)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, bool >::__type **isfinite** (_Tp)
- constexpr bool **isfinite** (double __x)
- constexpr bool **isfinite** (float __x)
- constexpr bool **isfinite** (long double __x)
- template<typename _CharT>
  bool isgraph (_CharT __c, const locale &__loc)

- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__enable_if<(__is_arithmetic< _Tp >::__value &&__is_arithmetic< _Up >::__value), bool >::__type **isgreater** (_Tp __x, _Up __y)
- constexpr bool **isgreater** (double __x, double __y)
- constexpr bool **isgreater** (float __x, float __y)
- constexpr bool **isgreater** (long double __x, long double __y)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__enable_if<(__is_arithmetic< _Tp >::__value &&__is_arithmetic< _Up >::__value), bool >::__type **isgreaterequal** (_Tp __x, _Up __y)
- constexpr bool **isgreaterequal** (double __x, double __y)
- constexpr bool **isgreaterequal** (float __x, float __y)
- constexpr bool **isgreaterequal** (long double __x, long double __y)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, bool >::__type **isinf** (_Tp)
- constexpr bool **isinf** (double __x)
- constexpr bool **isinf** (float __x)
- constexpr bool **isinf** (long double __x)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__enable_if<(__is_arithmetic< _Tp >::__value &&__is_arithmetic< _Up >::__value), bool >::__type **isless** (_Tp __x, _Up __y)
- constexpr bool **isless** (double __x, double __y)
- constexpr bool **isless** (float __x, float __y)
- constexpr bool **isless** (long double __x, long double __y)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__enable_if<(__is_arithmetic< _Tp >::__value &&__is_arithmetic< _Up >::__value), bool >::__type **islessequal** (_Tp __x, _Up __y)
- constexpr bool **islessequal** (double __x, double __y)
- constexpr bool **islessequal** (float __x, float __y)
- constexpr bool **islessequal** (long double __x, long double __y)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__enable_if<(__is_arithmetic< _Tp >::__value &&__is_arithmetic< _Up >::__value), bool >::__type **islessgreater** (_Tp __x, _Up __y)
- constexpr bool **islessgreater** (double __x, double __y)
- constexpr bool **islessgreater** (float __x, float __y)
- constexpr bool **islessgreater** (long double __x, long double __y)
- template<typename _CharT>
  bool islower (_CharT __c, const locale &__loc)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, bool >::__type **isnan** (_Tp)
- constexpr bool **isnan** (double __x)
- constexpr bool **isnan** (float __x)
- constexpr bool **isnan** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, bool >::__type **isnormal** (_Tp __x)
- constexpr bool **isnormal** (double __x)
- constexpr bool **isnormal** (float __x)
- constexpr bool **isnormal** (long double __x)
- template<typename _CharT>
  bool isprint (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool ispunct (_CharT __c, const locale &__loc)

- template<typename _CharT>
  bool isspace (_CharT __c, const locale &__loc)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__enable_if<(__is_arithmetic< _Tp >::__value &&__is_arithmetic< _Up >::__value), bool >::__type **isunordered** (_Tp __x, _Up __y)
- constexpr bool **isunordered** (double __x, double __y)
- constexpr bool **isunordered** (float __x, float __y)
- constexpr bool **isunordered** (long double __x, long double __y)
- template<typename _CharT>
  bool isupper (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool isxdigit (_CharT __c, const locale &__loc)
- template<typename _FIter1, typename _FIter2>
  constexpr void **iter_swap** (_FIter1, _FIter2)
- template<typename _ForwardIterator1, typename _ForwardIterator2>
  constexpr void iter_swap (_ForwardIterator1 __a, _ForwardIterator2 __b)
- template<typename _Tp>
  _Tp kill_dependency (_Tp __y) noexcept
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type laguerre (unsigned int __n, _Tp __x)
- float laguerref (unsigned int __n, float __x)
- long double laguerrel (unsigned int __n, long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **ldexp** (_Tp __x, int __exp)
- template<typename _Tp, typename _Abi, typename..., typename _Arg2 = _Extra_argument_type<int, _Tp, _Abi>, typename _R = _Math↩
  _return_type_t< decltype(std::ldexp(declval<double>(), _Arg2::declval())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **ldexp** (const simd< _Tp, _Abi > &__x, const typename _Arg2::type &__y)
- constexpr float **ldexp** (float __x, int __exp)
- constexpr long double **ldexp** (long double __x, int __exp)
- ios_base & left (ios_base &__base)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type legendre (unsigned int __l, _Tp __x)
- float legendref (unsigned int __l, float __x)
- long double legendrel (unsigned int __l, long double __x)
- template<typename _II1, typename _II2>
  constexpr bool lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)
- template<typename _II1, typename _II2, typename _Compare>
  constexpr bool lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, _Compare __comp)
- template<typename _IIter1, typename _IIter2>
  constexpr bool **lexicographical_compare** (_IIter1, _IIter1, _IIter2, _IIter2)
- template<typename _IIter1, typename _IIter2, typename _Compare>
  constexpr bool **lexicographical_compare** (_IIter1, _IIter1, _IIter2, _IIter2, _Compare)
- template<typename _InputIter1, typename _InputIter2>
  constexpr auto **lexicographical_compare_three_way** (_InputIter1 __first1, _InputIter1 __last1, _InputIter2 __↩
  first2, _InputIter2 __last2)
- template<typename _InputIter1, typename _InputIter2, typename _Comp>
  constexpr auto lexicographical_compare_three_way (_InputIter1 __first1, _InputIter1 __last1, _InputIter2 __first2, _InputIter2 __last2, _Comp __comp) -> decltype(__comp(*__first1, *__first2))
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **lgamma** (_Tp __x)
- constexpr float **lgamma** (float __x)

- constexpr long double **lgamma** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::lgamma(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **lgamma** (simd< _Tp, _Abi > __x)
- template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  list (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> list< _ValT, _Allocator >
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, longlong >::__type **llrint** (_Tp __x)
- constexpr long long **llrint** (float __x)
- constexpr long long **llrint** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::llrint(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **llrint** (simd< _Tp, _Abi > __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, longlong >::__type **llround** (_Tp __x)
- constexpr long long **llround** (float __x)
- constexpr long long **llround** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::llround(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **llround** (simd< _Tp, _Abi > __x)
- template<typename _L1, typename _L2, typename... _L3>
  void lock (_L1 &__l1, _L2 &__l2, _L3 &... __l3)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **log** (_Tp __x)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Log, _Expr, _Dom >, typename _Dom::value_type > **log** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  complex< _Tp > log (const complex< _Tp > &)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Log, _ValArray, _Tp >, _Tp > **log** (const valarray< _Tp > &__v)
- constexpr float **log** (float __x)
- constexpr long double **log** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::log(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **log** (simd< _Tp, _Abi > __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **log10** (_Tp __x)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Log10, _Expr, _Dom >, typename _Dom::value_type > **log10** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  complex< _Tp > log10 (const complex< _Tp > &)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Log10, _ValArray, _Tp >, _Tp > **log10** (const valarray< _Tp > &__v)
- constexpr float **log10** (float __x)
- constexpr long double **log10** (long double __x)

- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::log10(declval<double>())), ↩ _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **log10** (simd< _Tp, _Abi > __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **log1p** (_Tp __x)
- constexpr float **log1p** (float __x)
- constexpr long double **log1p** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::log1p(declval<double>())), ↩ _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **log1p** (simd< _Tp, _Abi > __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **log2** (_Tp __x)
- constexpr float **log2** (float __x)
- constexpr long double **log2** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::log2(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **log2** (simd< _Tp, _Abi > __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **logb** (_Tp __x)
- template<typename _Tp, typename _Abi, typename = __detail::__odr_helper>
  enable_if_t< is_floating_point< _Tp >::value, simd< _Tp, _Abi > > **logb** (const simd< _Tp, _Abi > &__x)
- constexpr float **logb** (float __x)
- constexpr long double **logb** (long double __x)
- template<typename _FIter, typename _Tp>
  constexpr _FIter **lower_bound** (_FIter, _FIter, const _Tp &)
- template<typename _FIter, typename _Tp, typename _Compare>
  constexpr _FIter **lower_bound** (_FIter, _FIter, const _Tp &, _Compare)
- template<typename _ForwardIterator, typename _Tp>
  constexpr _ForwardIterator lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)
- template<typename _ForwardIterator, typename _Tp, typename _Compare>
  constexpr _ForwardIterator lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, ↩ _Compare __comp)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, long >::__type **lrint** (_Tp __x)
- constexpr long **lrint** (float __x)
- constexpr long **lrint** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::lrint(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **lrint** (simd< _Tp, _Abi > __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, long >::__type **lround** (_Tp __x)
- constexpr long **lround** (float __x)
- constexpr long **lround** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::lround(declval<double>())), ↩ _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **lround** (simd< _Tp, _Abi > __x)
- error_code make_error_code (errc __e) noexcept

- error_code make_error_code (future_errc __errc) noexcept
- error_code **make_error_code** (io_errc __e) noexcept
- error_condition make_error_condition (errc __e) noexcept
- error_condition make_error_condition (future_errc __errc) noexcept
- error_condition **make_error_condition** (io_errc __e) noexcept
- template<typename _Ex>
  exception_ptr make_exception_ptr (_Ex __ex) noexcept
- template<typename _RAIter>
  constexpr void **make_heap** (_RAIter, _RAIter)
- template<typename _RAIter, typename _Compare>
  constexpr void **make_heap** (_RAIter, _RAIter, _Compare)
- template<typename _RandomAccessIterator>
  constexpr void make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _Iterator>
  constexpr move_iterator< _Iterator > **make_move_iterator** (_Iterator __i)
- template<typename _Tp, typename _Alloc, typename... _Args>
  constexpr _Tp **make_obj_using_allocator** (const _Alloc &__a, _Args &&... __args)
- template<typename _T1, typename _T2>
  constexpr pair< typename __decay_and_strip< _T1 >::__type, typename __decay_and_strip< _T2 >::__type > make_pair (_T1 &&__x, _T2 &&__y)
- template<typename _Iterator>
  constexpr reverse_iterator< _Iterator > make_reverse_iterator (_Iterator __i)
- template<typename... _Elements>
  constexpr tuple< typename __decay_and_strip< _Elements >::__type... > make_tuple (_Elements &&... __↩ args)
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  map (_InputIterator, _InputIterator, _Allocator) -> map< __iter_key_t< _InputIterator >, __iter_val_t< _Input↩ Iterator >, less< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Compare = less<__iter_key_t<_InputIterator>>, typename _Allocator = allocator<__iter↩ _to_alloc_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  map (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> map< __iter_key_t< ↩ _InputIterator >, __iter_val_t< _InputIterator >, _Compare, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  map (initializer_list< pair< _Key, _Tp > >, _Allocator) -> map< _Key, _Tp, less< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Compare = less<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  map (initializer_list< pair< _Key, _Tp > >, _Compare=_Compare(), _Allocator=_Allocator()) -> map< _Key, _Tp, _Compare, _Allocator >
- template<typename _Tp>
  constexpr const _Tp & max (const _Tp &__a, const _Tp &__b)
- template<typename _Tp, typename _Compare>
  constexpr const _Tp & max (const _Tp &__a, const _Tp &__b, _Compare __comp)
- template<typename _Tp>
  constexpr _Tp **max** (initializer_list< _Tp >)
- template<typename _Tp, typename _Compare>
  constexpr _Tp **max** (initializer_list< _Tp >, _Compare)
- template<typename _FIter>
  constexpr _FIter **max_element** (_FIter, _FIter)

- template<typename _FIter, typename _Compare>
  constexpr _FIter **max_element** (_FIter, _FIter, _Compare)
- template<typename _ForwardIterator>
  constexpr _ForwardIterator max_element (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _Compare>
  constexpr _ForwardIterator max_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)
- template<typename _Tp, typename _Class>
  constexpr _Mem_fn< _Tp _Class::* > mem_fn (_Tp _Class::*__pm) noexcept
- template<typename _Ret, typename _Tp>
  const_mem_fun_t< _Ret, _Tp > **mem_fun** (_Ret(_Tp::*__f)() const)
- template<typename _Ret, typename _Tp>
  mem_fun_t< _Ret, _Tp > **mem_fun** (_Ret(_Tp::*__f)())
- template<typename _Ret, typename _Tp, typename _Arg>
  const_mem_fun1_t< _Ret, _Tp, _Arg > **mem_fun** (_Ret(_Tp::*__f)(_Arg) const)
- template<typename _Ret, typename _Tp, typename _Arg>
  mem_fun1_t< _Ret, _Tp, _Arg > **mem_fun** (_Ret(_Tp::*__f)(_Arg))
- template<typename _Ret, typename _Tp>
  const_mem_fun_ref_t< _Ret, _Tp > **mem_fun_ref** (_Ret(_Tp::*__f)() const)
- template<typename _Ret, typename _Tp>
  mem_fun_ref_t< _Ret, _Tp > **mem_fun_ref** (_Ret(_Tp::*__f)())
- template<typename _Ret, typename _Tp, typename _Arg>
  const_mem_fun1_ref_t< _Ret, _Tp, _Arg > **mem_fun_ref** (_Ret(_Tp::*__f)(_Arg) const)
- template<typename _Ret, typename _Tp, typename _Arg>
  mem_fun1_ref_t< _Ret, _Tp, _Arg > **mem_fun_ref** (_Ret(_Tp::*__f)(_Arg))
- void * **memchr** (void *__s, int __c, size_t __n)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  constexpr _OIter **merge** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare>
  constexpr _OIter **merge** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>
  constexpr _OutputIterator merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _↩
  InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _↩
  InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _Tp>
  constexpr const _Tp & min (const _Tp &__a, const _Tp &__b)
- template<typename _Tp, typename _Compare>
  constexpr const _Tp & min (const _Tp &__a, const _Tp &__b, _Compare __comp)
- template<typename _Tp>
  constexpr _Tp **min** (initializer_list< _Tp >)
- template<typename _Tp, typename _Compare>
  constexpr _Tp **min** (initializer_list< _Tp >, _Compare)
- template<typename _FIter>
  constexpr _FIter **min_element** (_FIter, _FIter)
- template<typename _FIter, typename _Compare>
  constexpr _FIter **min_element** (_FIter, _FIter, _Compare)
- template<typename _ForwardIterator>
  constexpr _ForwardIterator min_element (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _Compare>
  constexpr _ForwardIterator min_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)
- template<typename _Tp>
  constexpr pair< const _Tp &, const _Tp & > minmax (const _Tp &__a, const _Tp &__b)

- template<typename _Tp, typename _Compare>
  constexpr pair< const _Tp &, const _Tp & > minmax (const _Tp &__a, const _Tp &__b, _Compare __comp)
- template<typename _Tp>
  constexpr pair< _Tp, _Tp > **minmax** (initializer_list< _Tp >)
- template<typename _Tp, typename _Compare>
  constexpr pair< _Tp, _Tp > **minmax** (initializer_list< _Tp >, _Compare)
- template<typename _FIter>
  constexpr pair< _FIter, _FIter > **minmax_element** (_FIter, _FIter)
- template<typename _FIter, typename _Compare>
  constexpr pair< _FIter, _FIter > **minmax_element** (_FIter, _FIter, _Compare)
- template<typename _ForwardIterator>
  constexpr pair< _ForwardIterator, _ForwardIterator > minmax_element (_ForwardIterator __first, _Forward↩
  Iterator __last)
- template<typename _ForwardIterator, typename _Compare>
  constexpr pair< _ForwardIterator, _ForwardIterator > minmax_element (_ForwardIterator __first, _Forward↩
  Iterator __last, _Compare __comp)
- template<typename _IIter1, typename _IIter2>
  constexpr pair< _IIter1, _IIter2 > **mismatch** (_IIter1, _IIter1, _IIter2)
- template<typename _IIter1, typename _IIter2, typename _BinaryPredicate>
  constexpr pair< _IIter1, _IIter2 > **mismatch** (_IIter1, _IIter1, _IIter2, _BinaryPredicate)
- template<typename _InputIterator1, typename _InputIterator2>
  constexpr pair< _InputIterator1, _InputIterator2 > mismatch (_InputIterator1 __first1, _InputIterator1 __last1, ↩
  _InputIterator2 __first2)
- template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>
  constexpr pair< _InputIterator1, _InputIterator2 > mismatch (_InputIterator1 __first1, _InputIterator1 __last1, ↩
  _InputIterator2 __first2, _BinaryPredicate __binary_pred)
- template<typename _InputIterator1, typename _InputIterator2>
  constexpr pair< _InputIterator1, _InputIterator2 > mismatch (_InputIterator1 __first1, _InputIterator1 __last1, ↩
  _InputIterator2 __first2, _InputIterator2 __last2)
- template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>
  constexpr pair< _InputIterator1, _InputIterator2 > mismatch (_InputIterator1 __first1, _InputIterator1 __last1, ↩
  _InputIterator2 __first2, _InputIterator2 __last2, _BinaryPredicate __binary_pred)
- template<typename _Tp, typename _Abi, typename = __detail::__odr_helper>
  enable_if_t< is_floating_point_v< _Tp >, simd< _Tp, _Abi > > **modf** (const simd< _Tp, _Abi > &__x, simd< _Tp, _Abi > *__iptr)
- float **modf** (float __x, float *__iptr)
- long double **modf** (long double __x, long double *__iptr)
- template<typename _II, typename _OI>
  constexpr _OI move (_II __first, _II __last, _OI __result)
- template<typename _Tp>
  constexpr std::remove_reference< _Tp >::type && move (_Tp &&__t) noexcept
- template<typename _BI1, typename _BI2>
  constexpr _BI2 move_backward (_BI1 __first, _BI1 __last, _BI2 __result)
- template<typename _Tp>
  constexpr __conditional_t< __move_if_noexcept_cond< _Tp >::value, const _Tp &, _Tp && > move_if_noexcept (_Tp &__x) noexcept
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩
  Allocator<_Allocator>>
  multimap (_InputIterator, _InputIterator, _Allocator) -> multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, less< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Compare = less<__iter_key_t<_InputIterator>>, typename _Allocator = allocator<__iter↩
  _to_alloc_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>

multimap (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> multimap< __iter↩ _key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Compare, _Allocator >

- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  multimap (initializer_list< pair< _Key, _Tp > >, _Allocator) -> multimap< _Key, _Tp, less< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Compare = less<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  multimap (initializer_list< pair< _Key, _Tp > >, _Compare=_Compare(), _Allocator=_Allocator()) -> multimap< _Key, _Tp, _Compare, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  multiset (_InputIterator, _InputIterator, _Allocator) -> multiset< typename iterator_traits< _InputIterator >↩ ::value_type, less< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Compare = less<typename iterator_traits<_InputIterator>::value_type>, typename _↩ Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  multiset (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> multiset< typename iterator_traits< _InputIterator >::value_type, _Compare, _Allocator >
- template<typename _Key, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  multiset (initializer_list< _Key >, _Allocator) -> multiset< _Key, less< _Key >, _Allocator >
- template<typename _Key, typename _Compare = less<_Key>, typename _Allocator = allocator<_Key>, typename = _RequireNot↩ Allocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  multiset (initializer_list< _Key >, _Compare=_Compare(), _Allocator=_Allocator()) -> multiset< _Key, _↩ Compare, _Allocator >
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **nearbyint** (_Tp __x)
- constexpr float **nearbyint** (float __x)
- constexpr long double **nearbyint** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::nearbyint(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **nearbyint** (simd< _Tp, _Abi > __x)
- template<typename _InputIterator>
  constexpr _InputIterator **next** (_InputIterator __x, typename iterator_traits< _InputIterator >::difference_type _↩ _n=1)
- template<typename _BidirectionalIterator>
  constexpr bool next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)
- template<typename _BidirectionalIterator, typename _Compare>
  constexpr bool next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)
- template<typename _BIter>
  constexpr bool **next_permutation** (_BIter, _BIter)
- template<typename _BIter, typename _Compare>
  constexpr bool **next_permutation** (_BIter, _BIter, _Compare)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type **nextafter** (_Tp __x, _Up __y)
- template<typename _Tp, typename _Abi, typename..., typename _Arg2 = _Extra_argument_type<_Tp, _Tp, _Abi>, typename _R = _↩ Math_return_type_t< decltype(std::nextafter(declval<double>(), _Arg2::declval())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **nextafter** (const simd< _Tp, _Abi > &__x, const typename _Arg2::type &__y)
- constexpr float **nextafter** (float __x, float __y)
- constexpr long double **nextafter** (long double __x, long double __y)

- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **nexttoward** (_Tp __x, long double __y)
- constexpr float **nexttoward** (float __x, long double __y)
- constexpr long double **nexttoward** (long double __x, long double __y)
- ios_base & noboolalpha (ios_base &__base)
- template<typename _IIter, typename _Predicate>
  constexpr bool **none_of** (_IIter, _IIter, _Predicate)
- template<typename _InputIterator, typename _Predicate>
  constexpr bool none_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _Tp>
  constexpr __gnu_cxx::__promote< _Tp >::__type **norm** (_Tp __x)
- template<typename _Tp>
  _Tp constexpr norm (const complex< _Tp > &)
- template<typename _Tp>
  constexpr _Tp norm (const complex< _Tp > &__z)
- ios_base & noshowbase (ios_base &__base)
- ios_base & noshowpoint (ios_base &__base)
- ios_base & noshowpos (ios_base &__base)
- ios_base & noskipws (ios_base &__base)
- template<typename _Predicate>
  constexpr unary_negate< _Predicate > not1 (const _Predicate &__pred)
- template<typename _Predicate>
  constexpr binary_negate< _Predicate > not2 (const _Predicate &__pred)
- void **notify_all_at_thread_exit** (condition_variable &, unique_lock< mutex >)
- ios_base & nounitbuf (ios_base &__base)
- ios_base & nouppercase (ios_base &__base)
- template<typename _RAIter>
  constexpr void **nth_element** (_RAIter, _RAIter, _RAIter)
- template<typename _RAIter, typename _Compare>
  constexpr void **nth_element** (_RAIter, _RAIter, _RAIter, _Compare)
- template<typename _RandomAccessIterator>
  constexpr void nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccess↩
  Iterator __last)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccess↩
  Iterator __last, _Compare __comp)
- ios_base & oct (ios_base &__base)
- template<class _Dom>
  _Expr< _BinClos< struct std::__not_equal_to, _Expr, _ValArray, _Dom, typename _Dom::value_type >, type-
  name __fun< struct std::__not_equal_to, typename _Dom::value_type >::result_type > **operator!=** (const _↩
  Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__not_equal_to, _Expr, _Constant, _Dom, typename _Dom::value_type >, type-
  name __fun< struct std::__not_equal_to, typename _Dom::value_type >::result_type > **operator!=** (const _↩
  Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__not_equal_to, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std↩
  ::__not_equal_to, typename _Dom1::value_type >::result_type > **operator!=** (const _Expr< _Dom1, typename
  _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<typename _StateT>
  bool **operator!=** (const fpos< _StateT > &__lhs, const fpos< _StateT > &__rhs)

- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  bool operator!= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)
- template<typename _Key, typename _Compare, typename _Alloc>
  bool operator!= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)
- template<typename _Tp, typename _Seq>
  bool operator!= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() != __y.base() } -> convertible_to<bool>; }
  constexpr bool **operator!=** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _Key, typename _Compare, typename _Alloc>
  bool operator!= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)
- template<typename _Tp, typename _Seq>
  bool operator!= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)
- template<class _Dom>
  _Expr< _BinClos< struct std::__not_equal_to, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__not_equal_to, typename _Dom::value_type >::result_type > **operator!=** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__not_equal_to, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__not_equal_to, typename _Dom::value_type >::result_type > **operator!=** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom>
  _Expr< _BinClos< struct std::__modulus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__modulus, typename _Dom::value_type >::result_type > **operator%** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__modulus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__modulus, typename _Dom::value_type >::result_type > **operator%** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__modulus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__modulus, typename _Dom1::value_type >::result_type > **operator%** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__modulus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__modulus, typename _Dom::value_type >::result_type > **operator%** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__modulus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__modulus, typename _Dom::value_type >::result_type > **operator%** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- constexpr _Ios_Fmtflags **operator&** (_Ios_Fmtflags __a, _Ios_Fmtflags __b) noexcept
- constexpr _Ios_Iostate **operator&** (_Ios_Iostate __a, _Ios_Iostate __b) noexcept
- constexpr _Ios_Openmode **operator&** (_Ios_Openmode __a, _Ios_Openmode __b) noexcept
- constexpr chars_format **operator&** (chars_format __lhs, chars_format __rhs) noexcept
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_and, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__bitwise_and, typename _Dom::value_type >::result_type > **operator&** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)

- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_and, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__bitwise_and, typename _Dom::value_type >::result_type > **operator&** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__bitwise_and, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__bitwise_and, typename _Dom1::value_type >::result_type > **operator&** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_and, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__bitwise_and, typename _Dom::value_type >::result_type > **operator&** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_and, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__bitwise_and, typename _Dom::value_type >::result_type > **operator&** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- constexpr launch **operator&** (launch __x, launch __y) noexcept
- constexpr memory_order **operator&** (memory_order __m, __memory_order_modifier __mod) noexcept
- template<class _Dom>
  _Expr< _BinClos< struct std::__logical_and, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__logical_and, typename _Dom::value_type >::result_type > **operator&&** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__logical_and, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__logical_and, typename _Dom::value_type >::result_type > **operator&&** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__logical_and, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__logical_and, typename _Dom1::value_type >::result_type > **operator&&** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__logical_and, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__logical_and, typename _Dom::value_type >::result_type > **operator&&** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__logical_and, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__logical_and, typename _Dom::value_type >::result_type > **operator&&** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- constexpr const _Ios_Fmtflags & **operator&=** (_Ios_Fmtflags &__a, _Ios_Fmtflags __b) noexcept
- constexpr const _Ios_Iostate & **operator&=** (_Ios_Iostate &__a, _Ios_Iostate __b) noexcept
- constexpr const _Ios_Openmode & **operator&=** (_Ios_Openmode &__a, _Ios_Openmode __b) noexcept
- constexpr chars_format & **operator&=** (chars_format &__lhs, chars_format __rhs) noexcept
- constexpr launch & **operator&=** (launch &__x, launch __y) noexcept
- template<class _Dom>
  _Expr< _BinClos< struct std::__multiplies, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__multiplies, typename _Dom::value_type >::result_type > **operator*** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__multiplies, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__multiplies, typename _Dom::value_type >::result_type > **operator*** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)

- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__multiplies, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__multiplies, typename _Dom1::value_type >::result_type > **operator*** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__multiplies, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__multiplies, typename _Dom::value_type >::result_type > **operator*** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__multiplies, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__multiplies, typename _Dom::value_type >::result_type > **operator*** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > **operator+** (_CharT __lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > operator+ (_CharT __lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > **operator+** (basic_string< _CharT, _Traits, _Alloc > &&__lhs, _CharT __rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > **operator+** (basic_string< _CharT, _Traits, _Alloc > &&__lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > **operator+** (basic_string< _CharT, _Traits, _Alloc > &&__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > **operator+** (basic_string< _CharT, _Traits, _Alloc > &&__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > **operator+** (const _CharT *__lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > operator+ (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<class _Dom>
  _Expr< _BinClos< struct std::__plus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _fun< struct std::__plus, typename _Dom::value_type >::result_type > **operator+** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__plus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _fun< struct std::__plus, typename _Dom::value_type >::result_type > **operator+** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__plus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__plus, typename _Dom1::value_type >::result_type > **operator+** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, _CharT __rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > **operator+** (const basic_string< _CharT, _Traits, _Alloc > &__lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)

- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > operator+ (const basic_string< _CharT, _Traits, _Alloc > &↩ __lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > operator+ (const basic_string< _CharT, _Traits, _Alloc > &↩ __lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _Tp>
  constexpr complex< _Tp > operator+ (const complex< _Tp > &__x)
- template<class _Dom>
  _Expr< _BinClos< struct std::__plus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename ↩ __fun< struct std::__plus, typename _Dom::value_type >::result_type > **operator+** (const typename _Dom↩ ::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__plus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _↩ _fun< struct std::__plus, typename _Dom::value_type >::result_type > **operator+** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Iterator>
  requires requires { { __x.base() + __n } -> same_as<_Iterator>; }
  constexpr move_iterator< _Iterator > **operator+** (typename move_iterator< _Iterator >::difference_type __n, const move_iterator< _Iterator > &__x)
- template<typename _Iterator>
  constexpr reverse_iterator< _Iterator > **operator+** (typename reverse_iterator< _Iterator >::difference_type ↩ __n, const reverse_iterator< _Iterator > &__x)
- template<class _Dom>
  _Expr< _BinClos< struct std::__minus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _↩ _fun< struct std::__minus, typename _Dom::value_type >::result_type > **operator-** (const _Expr< _Dom, type- name _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__minus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _↩ _fun< struct std::__minus, typename _Dom::value_type >::result_type > **operator-** (const _Expr< _Dom, type- name _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__minus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__minus, typename _Dom1::value_type >::result_type > **operator-** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<typename _Tp>
  constexpr complex< _Tp > operator- (const complex< _Tp > &__x)
- template<typename _IteratorL, typename _IteratorR>
  constexpr auto **operator-** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y) -> decltype(__x.base() - __y.base())
- template<typename _IteratorL, typename _IteratorR>
  constexpr auto **operator-** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y) -> decltype(__y.base() - __x.base())
- template<class _Dom>
  _Expr< _BinClos< struct std::__minus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__minus, typename _Dom::value_type >::result_type > **operator-** (const typename _Dom↩ ::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__minus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _↩ _fun< struct std::__minus, typename _Dom::value_type >::result_type > **operator-** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)

- template<class _Dom>
  _Expr< _BinClos< struct std::__divides, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__divides, typename _Dom::value_type >::result_type > **operator**/ (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__divides, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__divides, typename _Dom::value_type >::result_type > **operator**/ (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__divides, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__divides, typename _Dom1::value_type >::result_type > **operator**/ (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__divides, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__divides, typename _Dom::value_type >::result_type > **operator**/ (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__divides, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__divides, typename _Dom::value_type >::result_type > **operator**/ (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom>
  _Expr< _BinClos< struct std::__less, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__less, typename _Dom::value_type >::result_type > **operator**< (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__less, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__less, typename _Dom::value_type >::result_type > **operator**< (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__less, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__less, typename _Dom1::value_type >::result_type > **operator**< (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() < __y.base() } -> convertible_to<bool>; }
  constexpr bool **operator**< (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  bool operator< (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)
- template<typename _Key, typename _Compare, typename _Alloc>
  bool operator< (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)
- template<typename _Tp, typename _Seq>
  bool operator< (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() > __y.base() } -> convertible_to<bool>; }
  constexpr bool **operator**< (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _Key, typename _Compare, typename _Alloc>
  bool operator< (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)
- template<typename _Tp, typename _Seq>
  bool operator< (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)

- template<class _Dom>
  _Expr< _BinClos< struct std::__less, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename ↩
  __fun< struct std::__less, typename _Dom::value_type >::result_type > **operator**< (const typename _Dom↩
  ::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
  constexpr bool operator< (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp, typename _Dp>
  constexpr bool operator< (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<class _Dom>
  _Expr< _BinClos< struct std::__less, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _↩
  _fun< struct std::__less, typename _Dom::value_type >::result_type > **operator**< (const valarray< typename
  _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp, typename _Dp>
  constexpr bool operator< (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const error_code
  &__e)
- template<typename _Ostream, typename _Tp>
  __rvalue_stream_insertion_t< _Ostream, _Tp > operator<< (_Ostream &&__os, const _Tp &__x)
- template<typename _CharT, typename _Traits, typename _MoneyT>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, _Put_money<
  _MoneyT > __f)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, _Put_time< _↩
  CharT > __f)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, _Resetiosflags
  __f)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, _Setbase __f)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, _Setfill< _CharT
  > __f)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, _Setiosflags __f)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, _Setprecision
  __f)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, _Setw __f)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, basic_string_view<
  _CharT, _Traits > __str)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const
  __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str)
- template<typename _CharT, typename _Traits, typename _Alloc>
  basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const
  basic_string< _CharT, _Traits, _Alloc > &__str)
- template<typename _Tp, typename _CharT, class _Traits>
  basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const complex<
  _Tp > &__x)

- template<class _Dom>
  _Expr< _BinClos< struct std::__shift_left, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__shift_left, typename _Dom::value_type >::result_type > **operator**<< (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__shift_left, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__shift_left, typename _Dom::value_type >::result_type > **operator**<< (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__shift_left, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::_↩ _shift_left, typename _Dom1::value_type >::result_type > **operator**<< (const _Expr< _Dom1, typename _↩ Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__shift_left, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__shift_left, typename _Dom::value_type >::result_type > **operator**<< (const typename _↩ Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__shift_left, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__shift_left, typename _Dom::value_type >::result_type > **operator**<< (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template ostream & **operator**<< (ostream &, _Resetiosflags)
- template ostream & **operator**<< (ostream &, _Setbase)
- template ostream & **operator**<< (ostream &, _Setfill< char >)
- template ostream & **operator**<< (ostream &, _Setiosflags)
- template ostream & **operator**<< (ostream &, _Setprecision)
- template ostream & **operator**<< (ostream &, _Setw)
- template ostream & **operator**<< (ostream &, const complex< double > &)
- template ostream & **operator**<< (ostream &, const complex< float > &)
- template ostream & **operator**<< (ostream &, const complex< long double > &)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_int_distribution< _IntType > &)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_real_distribution< _RealType > &)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const binomial_distribution< _IntType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const chi_squared_distribution< _RealType > &__x)
- template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const discard_block_engine< _RandomNumberEngine, __p, __r > &__x)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const discrete_distribution< _IntType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const fisher_f_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const gamma_distribution< _RealType > &__x)

- template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const linear_congruential_engine< _UIntType, __a, __c, __m > &__lcr)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const lognormal_distribution< _RealType > &__x)
- template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _↩
  UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__x)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const negative_binomial_distribution< _IntType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const normal_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const piecewise_constant_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const piecewise_linear_distribution< _RealType > &__x)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const poisson_distribution< _IntType > &__x)
- template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const shuffle_order_engine< _RandomNumberEngine, __k > &__x)
- template<typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::bernoulli_distribution &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::cauchy_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::exponential_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::extreme_value_distribution< _RealType > &__x)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::geometric_distribution< _IntType > &__x)
- template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::weibull_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const student_t_distribution< _RealType > &__x)

- template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const subtract_with_carry_engine< _UIntType, __w, __s, __r > &__x)
- template wostream & **operator**<< (wostream &, _Resetiosflags)
- template wostream & **operator**<< (wostream &, _Setbase)
- template wostream & **operator**<< (wostream &, _Setfill< wchar_t >)
- template wostream & **operator**<< (wostream &, _Setiosflags)
- template wostream & **operator**<< (wostream &, _Setprecision)
- template wostream & **operator**<< (wostream &, _Setw)
- template wostream & **operator**<< (wostream &, const complex< double > &)
- template wostream & **operator**<< (wostream &, const complex< float > &)
- template wostream & **operator**<< (wostream &, const complex< long double > &)
- template<typename _Ch, typename _Tr, typename _Tp, _Lock_policy _Lp>
  std::basic_ostream< _Ch, _Tr > & operator<< (std::basic_ostream< _Ch, _Tr > &__os, const __shared_ptr< _Tp, _Lp > &__p)
- template<typename _CharT, typename _Traits, typename _Tp, typename _Dp>
  basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const unique_ptr< _Tp, _Dp > &__p)
- template<class _Dom>
  _Expr< _BinClos< struct std::__less_equal, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__less_equal, typename _Dom::value_type >::result_type > **operator**<= (const _Expr< _↩ Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__less_equal, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__less_equal, typename _Dom::value_type >::result_type > **operator**<= (const _Expr< _↩ Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__less_equal, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std↩ ::__less_equal, typename _Dom1::value_type >::result_type > **operator**<= (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __y.base() < __x.base() } -> convertible_to<bool>; }
  constexpr bool **operator**<= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  bool operator<= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _↩ Compare, _Alloc > &__y)
- template<typename _Key, typename _Compare, typename _Alloc>
  bool operator<= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)
- template<typename _Tp, typename _Seq>
  bool operator<= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() >= __y.base() } -> convertible_to<bool>; }
  constexpr bool **operator**<= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _Key, typename _Compare, typename _Alloc>
  bool operator<= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)
- template<typename _Tp, typename _Seq>
  bool operator<= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)
- template<class _Dom>
  _Expr< _BinClos< struct std::__less_equal, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__less_equal, typename _Dom::value_type >::result_type > **operator**<= (const typename ↩ _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)

- template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
  constexpr bool operator<= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp, typename _Dp>
  constexpr bool operator<= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<class _Dom>
  _Expr< _BinClos< struct std::__less_equal, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__less_equal, typename _Dom::value_type >::result_type > **operator**<**=** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp, typename _Dp>
  constexpr bool operator<= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)
- strong_ordering operator<=> (const error_code &__lhs, const error_code &__rhs) noexcept
- strong_ordering operator<=> (const error_condition &__lhs, const error_condition &__rhs) noexcept
- template<typename _CharT, typename _Traits>
  constexpr auto **operator**<**=**> (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept -> decltype(__detail::__char_traits_cmp_cat< _Traits >(0))
- template<typename _Tp, typename _Up, _Lock_policy _Lp>
  strong_ordering **operator**<**=**> (const __shared_ptr< _Tp, _Lp > &__a, const __shared_ptr< _Up, _Lp > &__b) noexcept
- template<typename _Tp, _Lock_policy _Lp>
  strong_ordering **operator**<**=**> (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept
- template<typename _Tp, size_t _Nm>
  constexpr __detail::__synth3way_t< _Tp > **operator**<**=**> (const array< _Tp, _Nm > &__a, const array< _Tp, _Nm > &__b)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr auto operator<=> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs) noexcept -> decltype(__detail::__char_traits_cmp_cat< _Traits >(0))
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr auto operator<=> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs) noexcept -> decltype(__detail::__char_traits_cmp_cat< _Traits >(0))
- template<typename _Tp, typename _Alloc>
  __detail::__synth3way_t< _Tp > operator<=> (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)
- template<typename _Tp, typename _Alloc>
  __detail::__synth3way_t< _Tp > operator<=> (const forward_list< _Tp, _Alloc > &__x, const forward_list< _Tp, _Alloc > &__y)
- template<typename _Tp, typename _Alloc>
  __detail::__synth3way_t< _Tp > operator<=> (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  __detail::__synth3way_t< pair< const _Key, _Tp > > operator<=> (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)
- template<three_way_comparable _Iterator>
  constexpr compare_three_way_result_t< _Iterator > **operator**<**=**> (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)
- template<typename _IteratorL, three_way_comparable_with< _IteratorL > _IteratorR>
  constexpr compare_three_way_result_t< _IteratorL, _IteratorR > **operator**<**=**> (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _Tp, three_way_comparable _Seq>
  compare_three_way_result_t< _Seq > **operator**<**=**> (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)
- template<three_way_comparable _Iterator>
  constexpr compare_three_way_result_t< _Iterator, _Iterator > **operator**<**=**> (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)

- template<typename _IteratorL, three_way_comparable_with< _IteratorL > _IteratorR>
  constexpr compare_three_way_result_t< _IteratorL, _IteratorR > **operator<=>** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _Tp, three_way_comparable _Seq>
  compare_three_way_result_t< _Seq > **operator<=>** (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)
- template<typename... _Tps, typename... _Ups>
  requires (sizeof...(_Tps) == sizeof...(_Ups)) && (requires { typename __detail::__synth3way_t<_Tps, _Ups>; } && ...)
  constexpr common_comparison_category_t< __detail::__synth3way_t< _Tps, _Ups >... > **operator<=>** (const tuple< _Tps... > &__t, const tuple< _Ups... > &__u)
- template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
  requires three_way_comparable_with<typename unique_ptr<_Tp, _Dp>::pointer, typename unique_ptr<_Up, _Ep>::pointer>
  constexpr compare_three_way_result_t< typename unique_ptr< _Tp, _Dp >::pointer, typename unique_ptr< _Up, _Ep >::pointer > **operator<=>** (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp, typename _Dp>
  requires three_way_comparable<typename unique_ptr<_Tp, _Dp>::pointer>
  constexpr compare_three_way_result_t< typename unique_ptr< _Tp, _Dp >::pointer > **operator<=>** (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<typename _Tp, typename _Alloc>
  constexpr __detail::__synth3way_t< _Tp > operator<=> (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)
- constexpr strong_ordering **operator<=>** (monostate, monostate) noexcept
- template<typename _T1, typename _T2>
  constexpr bool operator== (const allocator< _T1 > &, const allocator< _T2 > &) noexcept
- bool operator== (const error_code &__lhs, const error_code &__rhs) noexcept
- bool operator== (const error_code &__lhs, const error_condition &__rhs) noexcept
- bool operator== (const error_condition &__lhs, const error_condition &__rhs) noexcept
- template<typename _Bi_iter, typename _Alloc>
  bool operator== (const match_results< _Bi_iter, _Alloc > &__m1, const match_results< _Bi_iter, _Alloc > &__m2)
- template<typename _CharT, typename _Traits>
  constexpr bool **operator==** (basic_string_view< _CharT, _Traits > __x, type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept
- template<typename _Tp, _Lock_policy _Lp>
  bool **operator==** (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept
- template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>
  bool **operator==** (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept
- template<class _Dom>
  _Expr< _BinClos< struct std::__equal_to, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__equal_to, typename _Dom::value_type >::result_type > **operator==** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__equal_to, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__equal_to, typename _Dom::value_type >::result_type > **operator==** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__equal_to, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__equal_to, typename _Dom1::value_type >::result_type > **operator==** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<typename _Tp, std::size_t _Nm>
  constexpr bool **operator==** (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)

- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr bool operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr bool operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs) noexcept
- template<typename _Tp, typename _Alloc>
  bool operator== (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)
- template<typename _Tp, typename _Alloc>
  bool operator== (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)
- template<typename _StateT>
  bool operator== (const fpos< _StateT > &__lhs, const fpos< _StateT > &__rhs)
- template<typename _Res, typename... _Args>
  bool operator== (const function< _Res(_Args...)> &__f, nullptr_t) noexcept
- template<typename _CharT, typename _Traits>
  bool **operator==** (const istreambuf_iterator< _CharT, _Traits > &__a, const istreambuf_iterator< _CharT, _Traits > &__b)
- template<typename _Tp, typename _Alloc>
  bool operator== (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  bool operator== (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)
- template<typename _Iterator>
  constexpr bool **operator==** (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() == __y.base() } -> convertible_to<bool>; }
  constexpr bool **operator==** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  bool operator== (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)
- template<typename _Key, typename _Compare, typename _Alloc>
  bool operator== (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)
- template<typename _Tp, typename _Seq>
  bool operator== (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)
- template<typename _Iterator>
  requires requires { { __x.base() == __y.base() } -> convertible_to<bool>; }
  constexpr bool **operator==** (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() == __y.base() } -> convertible_to<bool>; }
  constexpr bool operator== (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &↩ __y)
- template<typename _Key, typename _Compare, typename _Alloc>
  bool operator== (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)
- template<typename _Tp, typename _Seq>
  bool operator== (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)
- template<typename _RealType>
  bool **operator==** (const std::normal_distribution< _RealType > &__d1, const std::normal_distribution< _Real↩ Type > &__d2)
- template<typename... _Tps, typename... _Ups>
  requires (sizeof...(_Tps) == sizeof...(_Ups)) && (requires (const _Tps& __t, const _Ups& __u) { { __t == __u } -> __detail::__boolean_testable; } && ...)
  constexpr bool **operator==** (const tuple< _Tps... > &__t, const tuple< _Ups... > &__u)

- template<class _Dom>
  _Expr< _BinClos< struct std::__equal_to, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__equal_to, typename _Dom::value_type >::result_type > **operator==** (const typename _↩ Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
  constexpr bool operator== (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp, typename _Dp>
  constexpr bool operator== (const unique_ptr< _Tp, _Dp > &__x, nullptr_t) noexcept
- template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc>
  bool **operator==** (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc>
  bool **operator==** (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<class _Value, class _Hash, class _Pred, class _Alloc>
  bool **operator==** (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)
- template<class _Value, class _Hash, class _Pred, class _Alloc>
  bool **operator==** (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)
- template<class _Dom>
  _Expr< _BinClos< struct std::__equal_to, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__equal_to, typename _Dom::value_type >::result_type > **operator==** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp, typename _Alloc>
  constexpr bool operator== (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)
- constexpr bool **operator==** (monostate, monostate) noexcept
- template<typename _OutA1, typename _OutA2, typename... _InA>
  bool **operator==** (const scoped_allocator_adaptor< _OutA1, _InA... > &__a, const scoped_allocator_adaptor< _OutA2, _InA... > &__b) noexcept
- template<class _Dom>
  _Expr< _BinClos< struct std::__greater, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename ↩ __fun< struct std::__greater, typename _Dom::value_type >::result_type > **operator>** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__greater, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__greater, typename _Dom::value_type >::result_type > **operator>** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__greater, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__↩ greater, typename _Dom1::value_type >::result_type > **operator>** (const _Expr< _Dom1, typename _Dom1↩ ::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __y.base() < __x.base() } -> convertible_to<bool>; }
  constexpr bool **operator>** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  bool operator> (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)
- template<typename _Key, typename _Compare, typename _Alloc>
  bool operator> (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)
- template<typename _Tp, typename _Seq>
  bool operator> (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)

- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() < __y.base() } -> convertible_to<bool>; }
  constexpr bool **operator>** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _Key, typename _Compare, typename _Alloc>
  bool operator> (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)
- template<typename _Tp, typename _Seq>
  bool operator> (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)
- template<class _Dom>
  _Expr< _BinClos< struct std::__greater, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__greater, typename _Dom::value_type >::result_type > **operator>** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
  constexpr bool operator> (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp, typename _Dp>
  constexpr bool operator> (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<class _Dom>
  _Expr< _BinClos< struct std::__greater, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__greater, typename _Dom::value_type >::result_type > **operator>** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp, typename _Dp>
  constexpr bool operator> (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)
- template<class _Dom>
  _Expr< _BinClos< struct std::__greater_equal, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__greater_equal, typename _Dom::value_type >::result_type > **operator>=** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__greater_equal, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__greater_equal, typename _Dom::value_type >::result_type > **operator>=** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__greater_equal, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__greater_equal, typename _Dom1::value_type >::result_type > **operator>=** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() < __y.base() } -> convertible_to<bool>; }
  constexpr bool **operator>=** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  bool operator>= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)
- template<typename _Key, typename _Compare, typename _Alloc>
  bool operator>= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)
- template<typename _Tp, typename _Seq>
  bool operator>= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() <= __y.base() } -> convertible_to<bool>; }
  constexpr bool **operator>=** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _Key, typename _Compare, typename _Alloc>
  bool operator>= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)
- template<typename _Tp, typename _Seq>
  bool operator>= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)

- template<class _Dom>
  _Expr< _BinClos< struct std::__greater_equal, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__greater_equal, typename _Dom::value_type >::result_type > **operator>=** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
  constexpr bool operator>= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp, typename _Dp>
  constexpr bool operator>= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<class _Dom>
  _Expr< _BinClos< struct std::__greater_equal, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__greater_equal, typename _Dom::value_type >::result_type > **operator>=** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp, typename _Dp>
  bool operator>= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)
- template<typename _Istream, typename _Tp>
  __rvalue_stream_extraction_t< _Istream, _Tp > operator>> (_Istream &&__is, _Tp &&__x)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str)
- template<typename _CharT, typename _Traits, typename _MoneyT>
  basic_istream< _CharT, _Traits > & **operator>>** (basic_istream< _CharT, _Traits > &__is, _Get_money< _MoneyT > __f)
- template<typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & **operator>>** (basic_istream< _CharT, _Traits > &__is, _Get_time< _CharT > __f)
- template<typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & **operator>>** (basic_istream< _CharT, _Traits > &__is, _Resetiosflags __f)
- template<typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & **operator>>** (basic_istream< _CharT, _Traits > &__is, _Setbase __f)
- template<typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & **operator>>** (basic_istream< _CharT, _Traits > &__is, _Setfill< _CharT > __f)
- template<typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & **operator>>** (basic_istream< _CharT, _Traits > &__is, _Setiosflags __f)
- template<typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & **operator>>** (basic_istream< _CharT, _Traits > &__is, _Setprecision __f)
- template<typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & **operator>>** (basic_istream< _CharT, _Traits > &__is, _Setw __f)
- template<typename _CharT, typename _Traits, typename _Alloc>
  basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str)
- template<typename _Tp, typename _CharT, class _Traits>
  basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, complex< _Tp > &__x)
- template<> basic_istream< char > & **operator>>** (basic_istream< char > &__is, basic_string< char > &__str)
- template<class _Dom>
  _Expr< _BinClos< struct std::__shift_right, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__shift_right, typename _Dom::value_type >::result_type > **operator>>** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__shift_right, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__shift_right, typename _Dom::value_type >::result_type > **operator>>** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)

- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__shift_right, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__shift_right, typename _Dom1::value_type >::result_type > **operator**>> (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__shift_right, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__shift_right, typename _Dom::value_type >::result_type > **operator**>> (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__shift_right, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__shift_right, typename _Dom::value_type >::result_type > **operator**>> (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template istream & **operator**>> (istream &, _Resetiosflags)
- template istream & **operator**>> (istream &, _Setbase)
- template istream & **operator**>> (istream &, _Setfill< char >)
- template istream & **operator**>> (istream &, _Setiosflags)
- template istream & **operator**>> (istream &, _Setprecision)
- template istream & **operator**>> (istream &, _Setw)
- template istream & **operator**>> (istream &, complex< double > &)
- template istream & **operator**>> (istream &, complex< float > &)
- template istream & **operator**>> (istream &, complex< long double > &)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_int_distribution< _IntType > &)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_real_distribution< _RealType > &)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, binomial_distribution< _IntType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, chi_squared_distribution< _RealType > &__x)
- template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, discard_block_engine< _RandomNumberEngine, __p, __r > &__x)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, discrete_distribution< _IntType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, fisher_f_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, gamma_distribution< _RealType > &__x)
- template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, linear_congruential_engine< _UIntType, __a, __c, __m > &__lcr)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, lognormal_distribution< _RealType > &__x)

- template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__x)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, negative_binomial_distribution< _IntType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, normal_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, piecewise_constant_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, piecewise_linear_distribution< _RealType > &__x)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, poisson_distribution< _IntType > &__x)
- template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, shuffle_order_engine< _RandomNumberEngine, __k > &__x)
- template<typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::bernoulli_distribution &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::cauchy_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::exponential_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::extreme_value_distribution< _RealType > &__x)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::geometric_distribution< _IntType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::weibull_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, student_t_distribution< _RealType > &__x)
- template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, subtract_with_carry_engine< _UIntType, __w, __s, __r > &__x)
- template wistream & **operator**>> (wistream &, _Resetiosflags)
- template wistream & **operator**>> (wistream &, _Setbase)
- template wistream & **operator**>> (wistream &, _Setfill< wchar_t >)
- template wistream & **operator**>> (wistream &, _Setiosflags)
- template wistream & **operator**>> (wistream &, _Setprecision)
- template wistream & **operator**>> (wistream &, _Setw)

- template wistream & **operator**>> (wistream &, complex< double > &)
- template wistream & **operator**>> (wistream &, complex< float > &)
- template wistream & **operator**>> (wistream &, complex< long double > &)
- constexpr _Ios_Fmtflags **operator**^ (_Ios_Fmtflags __a, _Ios_Fmtflags __b) noexcept
- constexpr _Ios_Iostate **operator**^ (_Ios_Iostate __a, _Ios_Iostate __b) noexcept
- constexpr _Ios_Openmode **operator**^ (_Ios_Openmode __a, _Ios_Openmode __b) noexcept
- constexpr chars_format **operator**^ (chars_format __lhs, chars_format __rhs) noexcept
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_xor, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__bitwise_xor, typename _Dom::value_type >::result_type > **operator**^ (const _Expr< _↩ Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_xor, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__bitwise_xor, typename _Dom::value_type >::result_type > **operator**^ (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__bitwise_xor, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std↩ ::__bitwise_xor, typename _Dom1::value_type >::result_type > **operator**^ (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_xor, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__bitwise_xor, typename _Dom::value_type >::result_type > **operator**^ (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_xor, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__bitwise_xor, typename _Dom::value_type >::result_type > **operator**^ (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- constexpr launch **operator**^ (launch __x, launch __y) noexcept
- constexpr const _Ios_Fmtflags & **operator**^= (_Ios_Fmtflags &__a, _Ios_Fmtflags __b) noexcept
- constexpr const _Ios_Iostate & **operator**^= (_Ios_Iostate &__a, _Ios_Iostate __b) noexcept
- constexpr const _Ios_Openmode & **operator**^= (_Ios_Openmode &__a, _Ios_Openmode __b) noexcept
- constexpr chars_format & **operator**^= (chars_format &__lhs, chars_format __rhs) noexcept
- constexpr launch & **operator**^= (launch &__x, launch __y) noexcept
- constexpr _Ios_Fmtflags **operator**| (_Ios_Fmtflags __a, _Ios_Fmtflags __b) noexcept
- constexpr _Ios_Iostate **operator**| (_Ios_Iostate __a, _Ios_Iostate __b) noexcept
- constexpr _Ios_Openmode **operator**| (_Ios_Openmode __a, _Ios_Openmode __b) noexcept
- constexpr chars_format **operator**| (chars_format __lhs, chars_format __rhs) noexcept
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_or, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__bitwise_or, typename _Dom::value_type >::result_type > **operator**| (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_or, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__bitwise_or, typename _Dom::value_type >::result_type > **operator**| (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__bitwise_or, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::↩ __bitwise_or, typename _Dom1::value_type >::result_type > **operator**| (const _Expr< _Dom1, typename _↩ Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_or, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__bitwise_or, typename _Dom::value_type >::result_type > **operator**| (const typename _↩ Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)

- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_or, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__bitwise_or, typename _Dom::value_type >::result_type > **operator**| (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- constexpr launch **operator**| (launch __x, launch __y) noexcept
- constexpr memory_order **operator**| (memory_order __m, __memory_order_modifier __mod) noexcept
- constexpr const _Ios_Fmtflags & **operator**|**=** (_Ios_Fmtflags &__a, _Ios_Fmtflags __b) noexcept
- constexpr const _Ios_Iostate & **operator**|**=** (_Ios_Iostate &__a, _Ios_Iostate __b) noexcept
- constexpr const _Ios_Openmode & **operator**|**=** (_Ios_Openmode &__a, _Ios_Openmode __b) noexcept
- constexpr chars_format & **operator**|**=** (chars_format &__lhs, chars_format __rhs) noexcept
- constexpr launch & **operator**|**=** (launch &__x, launch __y) noexcept
- template<class _Dom>
  _Expr< _BinClos< struct std::__logical_or, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__logical_or, typename _Dom::value_type >::result_type > **operator**|| (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__logical_or, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__logical_or, typename _Dom::value_type >::result_type > **operator**|| (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__logical_or, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__logical_or, typename _Dom1::value_type >::result_type > **operator**|| (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__logical_or, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__logical_or, typename _Dom::value_type >::result_type > **operator**|| (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__logical_or, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__logical_or, typename _Dom::value_type >::result_type > **operator**|| (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- constexpr _Ios_Fmtflags **operator**~ (_Ios_Fmtflags __a) noexcept
- constexpr _Ios_Iostate **operator**~ (_Ios_Iostate __a) noexcept
- constexpr _Ios_Openmode **operator**~ (_Ios_Openmode __a) noexcept
- constexpr chars_format **operator**~ (chars_format __fmt) noexcept
- constexpr launch **operator**~ (launch __x) noexcept
- template<typename _Fun, typename _Signature = __function_guide_t<_Fun, decltype(&_Fun::operator())>>
  packaged_task (_Fun) -> packaged_task< _Signature >
- template<typename _Res, typename... _ArgTypes>
  packaged_task (_Res(*)(_ArgTypes...)) -> packaged_task< _Res(_ArgTypes...)>
- template<typename _RAIter>
  constexpr void **partial_sort** (_RAIter, _RAIter, _RAIter)
- template<typename _RAIter, typename _Compare>
  constexpr void **partial_sort** (_RAIter, _RAIter, _RAIter, _Compare)
- template<typename _RandomAccessIterator>
  constexpr void partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp)
- template<typename _IIter, typename _RAIter>
  constexpr _RAIter **partial_sort_copy** (_IIter, _IIter, _RAIter, _RAIter)

- template<typename _IIter, typename _RAIter, typename _Compare>
  constexpr _RAIter **partial_sort_copy** (_IIter, _IIter, _RAIter, _RAIter, _Compare)
- template<typename _InputIterator, typename _RandomAccessIterator>
  constexpr _RandomAccessIterator partial_sort_copy (_InputIterator __first, _InputIterator __last, _Random↩
  AccessIterator __result_first, _RandomAccessIterator __result_last)
- template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare>
  constexpr _RandomAccessIterator partial_sort_copy (_InputIterator __first, _InputIterator __last, _Random↩
  AccessIterator __result_first, _RandomAccessIterator __result_last, _Compare __comp)
- template<typename _InputIterator, typename _OutputIterator>
  constexpr _OutputIterator partial_sum (_InputIterator __first, _InputIterator __last, _OutputIterator __result)
- template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation>
  constexpr _OutputIterator partial_sum (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _↩
  BinaryOperation __binary_op)
- template<typename _BIter, typename _Predicate>
  constexpr _BIter **partition** (_BIter, _BIter, _Predicate)
- template<typename _ForwardIterator, typename _Predicate>
  constexpr _ForwardIterator partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)
- template<typename _IIter, typename _OIter1, typename _OIter2, typename _Predicate>
  constexpr pair< _OIter1, _OIter2 > **partition_copy** (_IIter, _IIter, _OIter1, _OIter2, _Predicate)
- template<typename _InputIterator, typename _OutputIterator1, typename _OutputIterator2, typename _Predicate>
  constexpr pair< _OutputIterator1, _OutputIterator2 > partition_copy (_InputIterator __first, _InputIterator __last,
  _OutputIterator1 __out_true, _OutputIterator2 __out_false, _Predicate __pred)
- template<typename _FIter, typename _Predicate>
  constexpr _FIter **partition_point** (_FIter, _FIter, _Predicate)
- template<typename _ForwardIterator, typename _Predicate>
  constexpr _ForwardIterator partition_point (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)
- template<typename _Tp>
  complex< _Tp > polar (const _Tp &, const _Tp &=_Tp(0))
- template<typename _RAIter>
  constexpr void **pop_heap** (_RAIter, _RAIter)
- template<typename _RAIter, typename _Compare>
  constexpr void **pop_heap** (_RAIter, _RAIter, _Compare)
- template<typename _RandomAccessIterator>
  constexpr void pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type **pow** (_Tp __x, _Up __y)
- template<class _Dom>
  _Expr< _BinClos< struct std::_Pow, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _↩
  Dom::value_type > **pow** (const _Expr< _Dom, typename _Dom::value_type > &__e, const typename _Dom↩
  ::value_type &__t)
- template<class _Dom>
  _Expr< _BinClos< struct std::_Pow, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _↩
  Dom::value_type > **pow** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename
  _Dom::value_type > &__v)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::_Pow, _Expr, _Expr, _Dom1, _Dom2 >, typename _Dom1::value_type > **pow**
  (const _Expr< _Dom1, typename _Dom1::value_type > &__e1, const _Expr< _Dom2, typename _Dom2↩
  ::value_type > &__e2)
- template<typename _Tp>
  complex< _Tp > pow (const _Tp &, const complex< _Tp > &)

- template<typename _Tp, typename _Up>
  std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > **pow** (const _Tp &__x, const std::complex< _Up > &__y)
- template<typename _Tp>
  complex< _Tp > pow (const complex< _Tp > &, const _Tp &)
- template<typename _Tp>
  complex< _Tp > pow (const complex< _Tp > &, const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > pow (const complex< _Tp > &, int)
- template<typename _Tp, typename _Abi, typename..., typename _Arg2 = _Extra_argument_type<_Tp, _Tp, _Abi>, typename _R = _↩ Math_return_type_t< decltype(std::pow(declval<double>(), _Arg2::declval())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **pow** (const simd< _Tp, _Abi > &__x, const typename _Arg2::type &__y)
- template<typename _Tp, typename _Up>
  std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > **pow** (const std::complex< _Tp > &__x, const _Up &__y)
- template<typename _Tp, typename _Up>
  std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > **pow** (const std::complex< _Tp > &__x, const std::complex< _Up > &__y)
- template<class _Dom>
  _Expr< _BinClos< struct std::_Pow, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename ↩ _Dom::value_type > **pow** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom↩ ::value_type > &__e)
- template<typename _Tp>
  _Expr< _BinClos< struct std::_Pow, _Constant, _ValArray, _Tp, _Tp >, _Tp > **pow** (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)
- template<typename _Tp>
  _Expr< _BinClos< struct std::_Pow, _ValArray, _Constant, _Tp, _Tp >, _Tp > **pow** (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)
- template<typename _Tp>
  _Expr< _BinClos< struct std::_Pow, _ValArray, _ValArray, _Tp, _Tp >, _Tp > **pow** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::_Pow, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _↩ Dom::value_type > **pow** (const valarray< typename _Dom::valarray > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- constexpr float **pow** (float __x, float __y)
- constexpr long double **pow** (long double __x, long double __y)
- template<typename _BidirectionalIterator>
  constexpr _BidirectionalIterator **prev** (_BidirectionalIterator __x, typename iterator_traits< _BidirectionalIterator >::difference_type __n=1)
- template<typename _BidirectionalIterator>
  constexpr bool prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)
- template<typename _BidirectionalIterator, typename _Compare>
  constexpr bool prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)
- template<typename _BIter>
  constexpr bool **prev_permutation** (_BIter, _BIter)
- template<typename _BIter, typename _Compare>
  constexpr bool **prev_permutation** (_BIter, _BIter, _Compare)
- template<typename _Compare, typename _Container, typename = _RequireNotAllocator<_Compare>, typename = _RequireNot↩ Allocator<_Container>>
  priority_queue (_Compare, _Container) -> priority_queue< typename _Container::value_type, _Container, _↩ Compare >

- template<typename _Compare, typename _Container, typename _Allocator, typename = _RequireNotAllocator<_Compare>, typename = _RequireNotAllocator<_Container>>
  priority_queue (_Compare, _Container, _Allocator) -> priority_queue< typename _Container::value_type, _Container, _Compare >
- template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Compare = less<_ValT>, typename _Container = vector<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireNotAllocator<_Container>>
  priority_queue (_InputIterator, _InputIterator, _Compare=_Compare(), _Container=_Container()) -> priority_queue< _ValT, _Container, _Compare >
- template<typename _Tp>
  std::complex< typename __gnu_cxx::__promote< _Tp >::__type > **proj** (_Tp __x)
- template<typename _Tp>
  std::complex< _Tp > **proj** (const std::complex< _Tp > &)
- template<typename _Arg, typename _Result>
  pointer_to_unary_function< _Arg, _Result > ptr_fun (_Result(*__x)(_Arg))
- template<typename _Arg1, typename _Arg2, typename _Result>
  pointer_to_binary_function< _Arg1, _Arg2, _Result > ptr_fun (_Result(*__x)(_Arg1, _Arg2))
- template<typename _RAIter>
  constexpr void **push_heap** (_RAIter, _RAIter)
- template<typename _RAIter, typename _Compare>
  constexpr void **push_heap** (_RAIter, _RAIter, _Compare)
- template<typename _RandomAccessIterator>
  constexpr void push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _MoneyT>
  _Put_money< _MoneyT > put_money (const _MoneyT &__mon, bool __intl=false)
- template<typename _CharT>
  _Put_time< _CharT > put_time (const std::tm *__tmb, const _CharT *__fmt)
- template<typename _Container, typename = _RequireNotAllocator<_Container>>
  queue (_Container) -> queue< typename _Container::value_type, _Container >
- template<typename _Container, typename _Allocator, typename = _RequireNotAllocator<_Container>>
  queue (_Container, _Allocator) -> queue< typename _Container::value_type, _Container >
- template<typename _RAIter>
  void **random_shuffle** (_RAIter, _RAIter)
- template<typename _RAIter, typename _Generator>
  void **random_shuffle** (_RAIter, _RAIter, _Generator &&)
- template<typename _RandomAccessIterator>
  void random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _RandomNumberGenerator>
  void random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomNumberGenerator &&__rand)
- template<typename _Container>
  constexpr auto rbegin (_Container &__cont) noexcept(noexcept(__cont.rbegin())) -> decltype(__cont.rbegin())
- template<typename _Tp, size_t _Nm>
  constexpr reverse_iterator< _Tp * > rbegin (_Tp(&__arr)[_Nm]) noexcept
- template<typename _Container>
  constexpr auto rbegin (const _Container &__cont) noexcept(noexcept(__cont.rbegin())) -> decltype(__cont.rbegin())
- template<typename _Tp>
  constexpr reverse_iterator< const _Tp * > rbegin (initializer_list< _Tp > __il) noexcept
- template<typename _Tp>
  constexpr __gnu_cxx::__promote< _Tp >::__type **real** (_Tp __x)

- template<typename _Tp>
  constexpr _Tp **real** (const complex< _Tp > &__z)
- template<typename _InputIterator>
  constexpr iterator_traits< _InputIterator >::value_type reduce (_InputIterator __first, _InputIterator __last)
- template<typename _InputIterator, typename _Tp>
  constexpr _Tp reduce (_InputIterator __first, _InputIterator __last, _Tp __init)
- template<typename _InputIterator, typename _Tp, typename _BinaryOperation>
  constexpr _Tp reduce (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __binary_op)
- template<typename _Tp>
  reference_wrapper (_Tp &) -> reference_wrapper< _Tp >
- template<typename _Tp, typename _Tp1, _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > **reinterpret_pointer_cast** (const __shared_ptr< _Tp1, _Lp > &__r) noexcept
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type **remainder** (_Tp __x, _Up __y)
- template<typename _Tp, typename _Abi, typename..., typename _Arg2 = _Extra_argument_type<_Tp, _Tp, _Abi>, typename _R = _↩
  Math_return_type_t< decltype(std::remainder(declval<double>(), _Arg2::declval())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **remainder** (const simd< _Tp, _Abi > &__x, const typename _Arg2::type &__y)
- constexpr float **remainder** (float __x, float __y)
- constexpr long double **remainder** (long double __x, long double __y)
- template<typename _FIter, typename _Tp>
  constexpr _FIter **remove** (_FIter, _FIter, const _Tp &)
- template<typename _ForwardIterator, typename _Tp>
  constexpr _ForwardIterator remove (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)
- template<typename _IIter, typename _OIter, typename _Tp>
  constexpr _OIter **remove_copy** (_IIter, _IIter, _OIter, const _Tp &)
- template<typename _InputIterator, typename _OutputIterator, typename _Tp>
  constexpr _OutputIterator remove_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp &__value)
- template<typename _IIter, typename _OIter, typename _Predicate>
  constexpr _OIter **remove_copy_if** (_IIter, _IIter, _OIter, _Predicate)
- template<typename _InputIterator, typename _OutputIterator, typename _Predicate>
  constexpr _OutputIterator remove_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)
- template<typename _FIter, typename _Predicate>
  constexpr _FIter **remove_if** (_FIter, _FIter, _Predicate)
- template<typename _ForwardIterator, typename _Predicate>
  constexpr _ForwardIterator remove_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)
- template<typename _Tp, typename _Up>
  __gnu_cxx::__promote_2< _Tp, _Up >::__type **remquo** (_Tp __x, _Up __y, int *__pquo)
- template<typename _Tp, typename _Abi, typename..., typename _Arg2 = _Extra_argument_type<_Tp, _Tp, _Abi>, typename _Arg3 = _↩
  Extra_argument_type<int*, _Tp, _Abi>, typename _R = _Math_return_type_t< decltype(std::remquo(declval<double>(), _Arg2::declval(), _Arg3::declval())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **remquo** (const simd< _Tp, _Abi > &__x, const typename _Arg2::type &__y, const typename _Arg3::type &__z)
- float **remquo** (float __x, float __y, int *__pquo)
- long double **remquo** (long double __x, long double __y, int *__pquo)
- template<typename _Container>
  constexpr auto rend (_Container &__cont) noexcept(noexcept(__cont.rend())) -> decltype(__cont.rend())
- template<typename _Tp, size_t _Nm>
  constexpr reverse_iterator< _Tp * > rend (_Tp(&__arr)[_Nm]) noexcept
- template<typename _Container>
  constexpr auto rend (const _Container &__cont) noexcept(noexcept(__cont.rend())) -> decltype(__cont.rend())

- template<typename _Tp>
  constexpr reverse_iterator< const _Tp * > rend (initializer_list< _Tp > __il) noexcept
- template<typename _FIter, typename _Tp>
  constexpr void **replace** (_FIter, _FIter, const _Tp &, const _Tp &)
- template<typename _ForwardIterator, typename _Tp>
  constexpr void replace (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__old_value, const _Tp &__new_value)
- template<typename _IIter, typename _OIter, typename _Tp>
  constexpr _OIter **replace_copy** (_IIter, _IIter, _OIter, const _Tp &, const _Tp &)
- template<typename _InputIterator, typename _OutputIterator, typename _Tp>
  constexpr _OutputIterator replace_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp &__old_value, const _Tp &__new_value)
- template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _Tp>
  constexpr _OutputIterator replace_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred, const _Tp &__new_value)
- template<typename _Iter, typename _OIter, typename _Predicate, typename _Tp>
  constexpr _OIter **replace_copy_if** (_Iter, _Iter, _OIter, _Predicate, const _Tp &)
- template<typename _FIter, typename _Predicate, typename _Tp>
  constexpr void **replace_if** (_FIter, _FIter, _Predicate, const _Tp &)
- template<typename _ForwardIterator, typename _Predicate, typename _Tp>
  constexpr void replace_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, const _Tp &__new_value)
- _Resetiosflags resetiosflags (ios_base::fmtflags __mask)
- void rethrow_exception (exception_ptr)
- template<typename _Ex>
  void rethrow_if_nested (const _Ex &__ex)
- template<typename _Tp>
  void return_temporary_buffer (_Tp *__p)
- template<typename _BidirectionalIterator>
  constexpr void reverse (_BidirectionalIterator __first, _BidirectionalIterator __last)
- template<typename _BIter>
  constexpr void **reverse** (_BIter, _BIter)
- template<typename _BidirectionalIterator, typename _OutputIterator>
  constexpr _OutputIterator reverse_copy (_BidirectionalIterator __first, _BidirectionalIterator __last, _OutputIterator __result)
- template<typename _BIter, typename _OIter>
  constexpr _OIter **reverse_copy** (_BIter, _BIter, _OIter)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type riemann_zeta (_Tp __s)
- float riemann_zetaf (float __s)
- long double riemann_zetal (long double __s)
- ios_base & right (ios_base &__base)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **rint** (_Tp __x)
- constexpr float **rint** (float __x)
- constexpr long double **rint** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::rint(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **rint** (simd< _Tp, _Abi > __x)
- template<typename _FIter>
  constexpr _FIter **rotate** (_FIter, _FIter, _FIter)

- template<typename _ForwardIterator>
  constexpr _ForwardIterator rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last)
- template<typename _FIter, typename _OIter>
  constexpr _OIter **rotate_copy** (_FIter, _FIter, _FIter, _OIter)
- template<typename _ForwardIterator, typename _OutputIterator>
  constexpr _OutputIterator rotate_copy (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator _↩
  _last, _OutputIterator __result)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **round** (_Tp __x)
- constexpr float **round** (float __x)
- constexpr long double **round** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::round(declval<double>())), ↩
  _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **round** (simd< _Tp, _Abi > __x)
- template<typename _PopulationIterator, typename _SampleIterator, typename _Distance, typename _UniformRandomBitGenerator>
  _SampleIterator sample (_PopulationIterator __first, _PopulationIterator __last, _SampleIterator __out, _Distance __n, _UniformRandomBitGenerator &&__g)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **scalbln** (_Tp __x, long __ex)
- template<typename _Tp, typename _Abi, typename..., typename _Arg2 = _Extra_argument_type<long, _Tp, _Abi>, typename _R = _↩
  Math_return_type_t< decltype(std::scalbln(declval<double>(), _Arg2::declval())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **scalbln** (const simd< _Tp, _Abi > &__x, const typename _Arg2::type &__y)
- constexpr float **scalbln** (float __x, long __ex)
- constexpr long double **scalbln** (long double __x, long __ex)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **scalbn** (_Tp __x, int __ex)
- template<typename _Tp, typename _Abi, typename..., typename _Arg2 = _Extra_argument_type<int, _Tp, _Abi>, typename _R = _Math↩
  _return_type_t< decltype(std::scalbn(declval<double>(), _Arg2::declval())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **scalbn** (const simd< _Tp, _Abi > &__x, const typename _Arg2::type &__y)
- constexpr float **scalbn** (float __x, int __ex)
- constexpr long double **scalbn** (long double __x, int __ex)
- ios_base & scientific (ios_base &__base)
- template<typename _FIter1, typename _FIter2>
  constexpr _FIter1 **search** (_FIter1, _FIter1, _FIter2, _FIter2)
- template<typename _FIter1, typename _FIter2, typename _BinaryPredicate>
  constexpr _FIter1 **search** (_FIter1, _FIter1, _FIter2, _FIter2, _BinaryPredicate)
- template<typename _ForwardIterator, typename _Searcher>
  constexpr _ForwardIterator search (_ForwardIterator __first, _ForwardIterator __last, const _Searcher &__↩
  searcher)
- template<typename _ForwardIterator1, typename _ForwardIterator2>
  constexpr _ForwardIterator1 search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 ↩
  __first2, _ForwardIterator2 __last2)
- template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>
  constexpr _ForwardIterator1 search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 ↩
  __first2, _ForwardIterator2 __last2, _BinaryPredicate __predicate)
- template<typename _FIter, typename _Size, typename _Tp>
  constexpr _FIter **search_n** (_FIter, _FIter, _Size, const _Tp &)
- template<typename _FIter, typename _Size, typename _Tp, typename _BinaryPredicate>
  constexpr _FIter **search_n** (_FIter, _FIter, _Size, const _Tp &, _BinaryPredicate)

- template<typename _ForwardIterator, typename _Integer, typename _Tp>
  constexpr _ForwardIterator search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val)
- template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate>
  constexpr _ForwardIterator search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  set (_InputIterator, _InputIterator, _Allocator) -> set< typename iterator_traits< _InputIterator >::value_type, less< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Compare = less<typename iterator_traits<_InputIterator>::value_type>, typename _↩ Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  set (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> set< typename iterator_traits< _InputIterator >::value_type, _Compare, _Allocator >
- template<typename _Key, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  set (initializer_list< _Key >, _Allocator) -> set< _Key, less< _Key >, _Allocator >
- template<typename _Key, typename _Compare = less<_Key>, typename _Allocator = allocator<_Key>, typename = _RequireNot↩ Allocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  set (initializer_list< _Key >, _Compare=_Compare(), _Allocator=_Allocator()) -> set< _Key, _Compare, _↩ Allocator >
- template<typename _IIter1, typename _IIter2, typename _OIter>
  constexpr _OIter **set_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare>
  constexpr _OIter **set_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>
  constexpr _OutputIterator set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  constexpr _OIter **set_intersection** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare>
  constexpr _OIter **set_intersection** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>
  constexpr _OutputIterator set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __↩ first2, _InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __↩ first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- new_handler set_new_handler (new_handler) throw ()
- template<typename _IIter1, typename _IIter2, typename _OIter>
  constexpr _OIter **set_symmetric_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare>
  constexpr _OIter **set_symmetric_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>
  constexpr _OutputIterator set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, _Input↩ Iterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, _Input↩ Iterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- terminate_handler set_terminate (terminate_handler) noexcept

- unexpected_handler set_unexpected (unexpected_handler) noexcept
- template<typename _IIter1, typename _IIter2, typename _OIter>
  constexpr _OIter **set_union** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare>
  constexpr _OIter **set_union** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>
  constexpr _OutputIterator set_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, ↩
  _InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator set_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, ↩
  _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- _Setbase setbase (int __base)
- template<typename _CharT>
  _Setfill< _CharT > setfill (_CharT __c)
- _Setiosflags setiosflags (ios_base::fmtflags __mask)
- _Setprecision setprecision (int __n)
- _Setw setw (int __n)
- template<typename _Tp, typename _Del>
  shared_ptr (unique_ptr< _Tp, _Del >) -> shared_ptr< _Tp >
- template<typename _Tp>
  shared_ptr (weak_ptr< _Tp >) -> shared_ptr< _Tp >
- template<typename _ForwardIterator>
  constexpr _ForwardIterator **shift_left** (_ForwardIterator __first, _ForwardIterator __last, typename iterator_traits< _ForwardIterator >::difference_type __n)
- template<typename _ForwardIterator>
  constexpr _ForwardIterator **shift_right** (_ForwardIterator __first, _ForwardIterator __last, typename iterator_traits< _ForwardIterator >::difference_type __n)
- ios_base & showbase (ios_base &__base)
- ios_base & showpoint (ios_base &__base)
- ios_base & showpos (ios_base &__base)
- template<typename _RAIter, typename _UGenerator>
  void **shuffle** (_RAIter, _RAIter, _UGenerator &&)
- template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator>
  void shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _UniformRandomNumberGenerator &&__g)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, bool >::__type **signbit** (_Tp __x)
- constexpr bool **signbit** (double __x)
- constexpr bool **signbit** (float __x)
- constexpr bool **signbit** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **sin** (_Tp __x)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Sin, _Expr, _Dom >, typename _Dom::value_type > **sin** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  complex< _Tp > sin (const complex< _Tp > &)
- template<typename _Tp, typename _Abi, typename = __detail::__odr_helper>
  enable_if_t< is_floating_point_v< _Tp >, simd< _Tp, _Abi > > **sin** (const simd< _Tp, _Abi > &__x)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Sin, _ValArray, _Tp >, _Tp > **sin** (const valarray< _Tp > &__v)
- constexpr float **sin** (float __x)

- constexpr long double **sin** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **sinh** (_Tp __x)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Sinh, _Expr, _Dom >, typename _Dom::value_type > **sinh** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  complex< _Tp > sinh (const complex< _Tp > &)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Sinh, _ValArray, _Tp >, _Tp > **sinh** (const valarray< _Tp > &__v)
- constexpr float **sinh** (float __x)
- constexpr long double **sinh** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::sinh(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **sinh** (simd< _Tp, _Abi > __x)
- template<typename _Container>
  constexpr auto size (const _Container &__cont) noexcept(noexcept(__cont.size())) -> decltype(__cont.size())
- template<typename _Tp, size_t _Nm>
  constexpr size_t size (const _Tp(&)[_Nm]) noexcept
- ios_base & skipws (ios_base &__base)
- template<typename _RAIter>
  constexpr void **sort** (_RAIter, _RAIter)
- template<typename _RAIter, typename _Compare>
  constexpr void **sort** (_RAIter, _RAIter, _Compare)
- template<typename _RandomAccessIterator>
  constexpr void sort (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _RAIter>
  constexpr void **sort_heap** (_RAIter, _RAIter)
- template<typename _RAIter, typename _Compare>
  constexpr void **sort_heap** (_RAIter, _RAIter, _Compare)
- template<typename _RandomAccessIterator>
  constexpr void sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type sph_bessel (unsigned int __n, _Tp __x)
- float sph_besself (unsigned int __n, float __x)
- long double sph_bessell (unsigned int __n, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type sph_legendre (unsigned int __l, unsigned int __m, _Tp __theta)
- float sph_legendref (unsigned int __l, unsigned int __m, float __theta)
- long double sph_legendrel (unsigned int __l, unsigned int __m, long double __theta)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type sph_neumann (unsigned int __n, _Tp __x)
- float sph_neumannf (unsigned int __n, float __x)
- long double sph_neumannl (unsigned int __n, long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **sqrt** (_Tp __x)

- template<class _Dom>
  _Expr< _UnClos< struct std::_Sqrt, _Expr, _Dom >, typename _Dom::value_type > **sqrt** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  complex< _Tp > sqrt (const complex< _Tp > &)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Sqrt, _ValArray, _Tp >, _Tp > **sqrt** (const valarray< _Tp > &__v)
- constexpr float **sqrt** (float __x)
- constexpr long double **sqrt** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::sqrt(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **sqrt** (simd< _Tp, _Abi > __x)
- template<typename _Container>
  constexpr auto **ssize** (const _Container &__cont) noexcept(noexcept(__cont.size())) -> common_type_t< ptrdiff_t, make_signed_t< decltype(__cont.size())> >
- template<typename _Tp, ptrdiff_t _Num>
  constexpr ptrdiff_t **ssize** (const _Tp(&)[_Num]) noexcept
- template<typename _BIter, typename _Predicate>
  _BIter **stable_partition** (_BIter, _BIter, _Predicate)
- template<typename _ForwardIterator, typename _Predicate>
  _ForwardIterator stable_partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)
- template<typename _RAIter>
  void **stable_sort** (_RAIter, _RAIter)
- template<typename _RAIter, typename _Compare>
  void **stable_sort** (_RAIter, _RAIter, _Compare)
- template<typename _RandomAccessIterator>
  void stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _Compare>
  void stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _Container, typename = _RequireNotAllocator<_Container>>
  stack (_Container) -> stack< typename _Container::value_type, _Container >
- template<typename _Container, typename _Allocator, typename = _RequireNotAllocator<_Container>>
  stack (_Container, _Allocator) -> stack< typename _Container::value_type, _Container >
- template<typename _Tp, typename _Tp1, _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > static_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept
- double **stod** (const string &__str, size_t *__idx=0)
- double **stod** (const wstring &__str, size_t *__idx=0)
- float **stof** (const string &__str, size_t *__idx=0)
- float **stof** (const wstring &__str, size_t *__idx=0)
- int **stoi** (const string &__str, size_t *__idx=0, int __base=10)
- int **stoi** (const wstring &__str, size_t *__idx=0, int __base=10)
- long **stol** (const string &__str, size_t *__idx=0, int __base=10)
- long **stol** (const wstring &__str, size_t *__idx=0, int __base=10)
- long double **stold** (const string &__str, size_t *__idx=0)
- long double **stold** (const wstring &__str, size_t *__idx=0)
- long long **stoll** (const string &__str, size_t *__idx=0, int __base=10)
- long long **stoll** (const wstring &__str, size_t *__idx=0, int __base=10)
- unsigned long **stoul** (const string &__str, size_t *__idx=0, int __base=10)
- unsigned long **stoul** (const wstring &__str, size_t *__idx=0, int __base=10)
- unsigned long long **stoull** (const string &__str, size_t *__idx=0, int __base=10)
- unsigned long long **stoull** (const wstring &__str, size_t *__idx=0, int __base=10)

- char * **strchr** (char *__s, int __n)
- char * **strpbrk** (char *__s1, const char *__s2)
- char * **strrchr** (char *__s, int __n)
- char * **strstr** (char *__s1, const char *__s2)
- template<typename _Ch_type, typename _Rx_traits>
  void swap (basic_regex< _Ch_type, _Rx_traits > &__lhs, basic_regex< _Ch_type, _Rx_traits > &__rhs) noexcept
- template<typename _Bi_iter, typename _Alloc>
  void swap (match_results< _Bi_iter, _Alloc > &__lhs, match_results< _Bi_iter, _Alloc > &__rhs) noexcept
- template<typename _Tp, _Lock_policy _Lp>
  void **swap** (__shared_ptr< _Tp, _Lp > &__a, __shared_ptr< _Tp, _Lp > &__b) noexcept
- template<typename _Tp, _Lock_policy _Lp>
  void **swap** (__weak_ptr< _Tp, _Lp > &__a, __weak_ptr< _Tp, _Lp > &__b) noexcept
- template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc>
  void **swap** (_Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)
- template<typename _Tp>
  constexpr _Require< __not_< __is_tuple_like< _Tp > >, is_move_constructible< _Tp >, is_move_assignable< _Tp > > swap (_Tp &, _Tp &) noexcept(__and_< is_nothrow_move_constructible< _Tp >, is_nothrow_move_assignable< _Tp > >::value)
- template<typename _Tp>
  requires (! __is_tuple_like<_Tp>::value) && is_move_constructible_v<_Tp> && is_move_assignable_v<_Tp>
  constexpr void swap (_Tp &__a, _Tp &__b) noexcept(/*conditional */) is_nothrow_move_assignable< _Tp > >
- template<typename _Tp, size_t _Nm>
  requires is_swappable_v<_Tp>
  constexpr void swap (_Tp(&__a)[_Nm], _Tp(&__b)[_Nm]) noexcept(/*conditional */)
- template<typename _Tp, size_t _Nm>
  constexpr __enable_if_t< __is_swappable< _Tp >::value > swap (_Tp(&__a)[_Nm], _Tp(&__b)[_Nm]) noexcept(__is_nothrow_swappable< _Tp >::value)
- template<typename _Tp, std::size_t _Nm>
  __enable_if_t<!__array_traits< _Tp, _Nm >::_Is_swappable::value > **swap** (array< _Tp, _Nm > &, array< _Tp, _Nm > &)=delete
- template<typename _Tp, std::size_t _Nm>
  constexpr __enable_if_t< __array_traits< _Tp, _Nm >::_Is_swappable::value > **swap** (array< _Tp, _Nm > &__one, array< _Tp, _Nm > &__two) noexcept(noexcept(__one.swap(__two)))
- template<class _CharT, class _Traits>
  void swap (basic_filebuf< _CharT, _Traits > &__x, basic_filebuf< _CharT, _Traits > &__y)
- template<class _CharT, class _Traits>
  void swap (basic_fstream< _CharT, _Traits > &__x, basic_fstream< _CharT, _Traits > &__y)
- template<class _CharT, class _Traits>
  void swap (basic_ifstream< _CharT, _Traits > &__x, basic_ifstream< _CharT, _Traits > &__y)
- template<class _CharT, class _Traits, class _Allocator>
  void swap (basic_istringstream< _CharT, _Traits, _Allocator > &__x, basic_istringstream< _CharT, _Traits, _Allocator > &__y)
- template<class _CharT, class _Traits>
  void swap (basic_ofstream< _CharT, _Traits > &__x, basic_ofstream< _CharT, _Traits > &__y)
- template<class _CharT, class _Traits, class _Allocator>
  void swap (basic_ostringstream< _CharT, _Traits, _Allocator > &__x, basic_ostringstream< _CharT, _Traits, _Allocator > &__y)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr void swap (basic_string< _CharT, _Traits, _Alloc > &__lhs, basic_string< _CharT, _Traits, _Alloc > &__rhs) noexcept(/*conditional */)

- template<class _CharT, class _Traits, class _Allocator>
  void swap (basic_stringbuf< _CharT, _Traits, _Allocator > &__x, basic_stringbuf< _CharT, _Traits, _Allocator > &__y) noexcept(noexcept(__x.swap(__y)))
- template<class _CharT, class _Traits, class _Allocator>
  void swap (basic_stringstream< _CharT, _Traits, _Allocator > &__x, basic_stringstream< _CharT, _Traits, _Allocator > &__y)
- template<typename _Tp, typename _Alloc>
  void swap (deque< _Tp, _Alloc > &__x, deque< _Tp, _Alloc > &__y) noexcept(/*conditional */)
- void **swap** (exception_ptr &__lhs, exception_ptr &__rhs)
- template<typename _Tp, typename _Alloc>
  void swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly) noexcept(noexcept(__lx.swap(__ly)))
- template<typename _Res, typename... _Args>
  void swap (function< _Res(_Args...)> &__x, function< _Res(_Args...)> &__y) noexcept
- template<typename _Tp, typename _Alloc>
  void swap (list< _Tp, _Alloc > &__x, list< _Tp, _Alloc > &__y) noexcept(/*conditional */)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  void swap (map< _Key, _Tp, _Compare, _Alloc > &__x, map< _Key, _Tp, _Compare, _Alloc > &__y) noexcept(/*conditional */)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  void swap (multimap< _Key, _Tp, _Compare, _Alloc > &__x, multimap< _Key, _Tp, _Compare, _Alloc > &__y) noexcept(/*conditional */)
- template<typename _Key, typename _Compare, typename _Alloc>
  void swap (multiset< _Key, _Compare, _Alloc > &__x, multiset< _Key, _Compare, _Alloc > &__y) noexcept(/*conditional */)
- template<typename _Res, typename... _ArgTypes>
  void swap (packaged_task< _Res(_ArgTypes...)> &__x, packaged_task< _Res(_ArgTypes...)> &__y) noexcept
- template<typename _Tp, typename _Sequence, typename _Compare>
  enable_if< __and_< __is_swappable< _Sequence >, __is_swappable< _Compare > >::value >::type **swap** (priority_queue< _Tp, _Sequence, _Compare > &__x, priority_queue< _Tp, _Sequence, _Compare > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename _Res>
  void **swap** (promise< _Res > &__x, promise< _Res > &__y) noexcept
- template<typename _Tp, typename _Seq>
  enable_if< __is_swappable< _Seq >::value >::type **swap** (queue< _Tp, _Seq > &__x, queue< _Tp, _Seq > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename _Key, typename _Compare, typename _Alloc>
  void swap (set< _Key, _Compare, _Alloc > &__x, set< _Key, _Compare, _Alloc > &__y) noexcept(/*conditional */)
- template<typename _Tp, typename _Seq>
  enable_if< __is_swappable< _Seq >::value >::type **swap** (stack< _Tp, _Seq > &__x, stack< _Tp, _Seq > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename... _Elements>
  constexpr enable_if<!__and_< __is_swappable< _Elements >... >::value >::type swap (tuple< _Elements... > &, tuple< _Elements... > &)=delete
- template<typename... _Elements>
  constexpr enable_if< __and_< __is_swappable< _Elements >... >::value >::type swap (tuple< _Elements... > &__x, tuple< _Elements... > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename _Tp, typename _Dp>
  enable_if<!__is_swappable< _Dp >::value >::type **swap** (unique_ptr< _Tp, _Dp > &, unique_ptr< _Tp, _Dp > &)=delete
- template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc>
  void **swap** (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))

- template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc>
  void **swap** (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))
- template<class _Value, class _Hash, class _Pred, class _Alloc>
  void **swap** (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))
- template<class _Value, class _Hash, class _Pred, class _Alloc>
  void **swap** (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, unordered_set< _Value, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename _Tp, typename _Alloc>
  constexpr void swap (vector< _Tp, _Alloc > &__x, vector< _Tp, _Alloc > &__y) noexcept(/*conditional */)
- void **swap** (thread &__x, thread &__y) noexcept
- template<typename _Mutex>
  void swap (unique_lock< _Mutex > &__x, unique_lock< _Mutex > &__y) noexcept
- template<typename _Tp, typename _Dp>
  constexpr enable_if< __is_swappable< _Dp >::value >::type swap (unique_ptr< _Tp, _Dp > &__x, unique_ptr< _Tp, _Dp > &__y) noexcept
- template<typename _Tp>
  void swap (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b) noexcept
- template<typename _FIter1, typename _FIter2>
  constexpr _FIter2 **swap_ranges** (_FIter1, _FIter1, _FIter2)
- template<typename _ForwardIterator1, typename _ForwardIterator2>
  constexpr _ForwardIterator2 swap_ranges (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)
- const error_category & system_category () noexcept
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **tan** (_Tp __x)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Tan, _Expr, _Dom >, typename _Dom::value_type > **tan** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  complex< _Tp > tan (const complex< _Tp > &)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Tan, _ValArray, _Tp >, _Tp > **tan** (const valarray< _Tp > &__v)
- constexpr float **tan** (float __x)
- constexpr long double **tan** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::tan(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **tan** (simd< _Tp, _Abi > __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **tanh** (_Tp __x)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Tanh, _Expr, _Dom >, typename _Dom::value_type > **tanh** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  complex< _Tp > tanh (const complex< _Tp > &)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Tanh, _ValArray, _Tp >, _Tp > **tanh** (const valarray< _Tp > &__v)
- constexpr float **tanh** (float __x)
- constexpr long double **tanh** (long double __x)

- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::tanh(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **tanh** (simd< _Tp, _Abi > __x)
- void terminate () noexcept
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **tgamma** (_Tp __x)
- constexpr float **tgamma** (float __x)
- constexpr long double **tgamma** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::tgamma(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **tgamma** (simd< _Tp, _Abi > __x)
- template<typename _Tp>
  void throw_with_nested (_Tp &&__t)
- template<typename... _Elements>
  constexpr tuple< _Elements &... > tie (_Elements &... __args) noexcept
- template<typename _Tp>
  constexpr _Tp * to_address (_Tp *__ptr) noexcept
- template<typename _Ptr>
  constexpr auto to_address (const _Ptr &__ptr) noexcept
- to_chars_result **to_chars** (char *, char *, bool, int=10)=delete
- constexpr to_chars_result **to_chars** (char *__first, char *__last, char __value, int __base=10)
- constexpr to_chars_result **to_chars** (char *__first, char *__last, signed char __value, int __base=10)
- constexpr to_chars_result **to_chars** (char *__first, char *__last, signed int __value, int __base=10)
- constexpr to_chars_result **to_chars** (char *__first, char *__last, signed long __value, int __base=10)
- constexpr to_chars_result **to_chars** (char *__first, char *__last, signed long long __value, int __base=10)
- constexpr to_chars_result **to_chars** (char *__first, char *__last, signed short __value, int __base=10)
- constexpr to_chars_result **to_chars** (char *__first, char *__last, unsigned char __value, int __base=10)
- constexpr to_chars_result **to_chars** (char *__first, char *__last, unsigned int __value, int __base=10)
- constexpr to_chars_result **to_chars** (char *__first, char *__last, unsigned long __value, int __base=10)
- constexpr to_chars_result **to_chars** (char *__first, char *__last, unsigned long long __value, int __base=10)
- constexpr to_chars_result **to_chars** (char *__first, char *__last, unsigned short __value, int __base=10)
- string **to_string** (double __val)
- string **to_string** (float __val)
- string **to_string** (int __val) noexcept
- string **to_string** (long __val) noexcept
- string **to_string** (long double __val)
- string **to_string** (long long __val)
- string **to_string** (unsigned __val) noexcept
- string **to_string** (unsigned long __val) noexcept
- string **to_string** (unsigned long long __val)
- wstring **to_wstring** (double __val)
- wstring **to_wstring** (float __val)
- wstring **to_wstring** (int __val)
- wstring **to_wstring** (long __val)
- wstring **to_wstring** (long double __val)
- wstring **to_wstring** (long long __val)
- wstring **to_wstring** (unsigned __val)
- wstring **to_wstring** (unsigned long __val)
- wstring **to_wstring** (unsigned long long __val)

- template<typename _CharT>
  _CharT tolower (_CharT __c, const locale &__loc)
- template<typename _CharT>
  _CharT toupper (_CharT __c, const locale &__loc)
- template<typename _IIter, typename _OIter, typename _UnaryOperation>
  constexpr _OIter **transform** (_IIter, _IIter, _OIter, _UnaryOperation)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _BinaryOperation>
  constexpr _OIter **transform** (_IIter1, _IIter1, _IIter2, _OIter, _BinaryOperation)
- template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation>
  constexpr _OutputIterator transform (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _↩
  UnaryOperation __unary_op)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _BinaryOperation>
  constexpr _OutputIterator transform (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _↩
  OutputIterator __result, _BinaryOperation __binary_op)
- template<typename _InputIterator, typename _OutputIterator, typename _Tp, typename _BinaryOperation, typename _UnaryOperation>
  constexpr _OutputIterator transform_exclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Tp __init, _BinaryOperation __binary_op, _UnaryOperation __unary_op)
- template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation, typename _UnaryOperation>
  constexpr _OutputIterator transform_inclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op, _UnaryOperation __unary_op)
- template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation, typename _UnaryOperation, typename _Tp>
  constexpr _OutputIterator transform_inclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op, _UnaryOperation __unary_op, _Tp __init)
- template<typename _InputIterator, typename _Tp, typename _BinaryOperation, typename _UnaryOperation>
  constexpr _Tp transform_reduce (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __↩
  binary_op, _UnaryOperation __unary_op)
- template<typename _InputIterator1, typename _InputIterator2, typename _Tp>
  constexpr _Tp transform_reduce (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init)
- template<typename _InputIterator1, typename _InputIterator2, typename _Tp, typename _BinaryOperation1, typename _Binary↩
  Operation2>
  constexpr _Tp transform_reduce (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init, _BinaryOperation1 __binary_op1, _BinaryOperation2 __binary_op2)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **trunc** (_Tp __x)
- constexpr float **trunc** (float __x)
- constexpr long double **trunc** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::trunc(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **trunc** (simd< _Tp, _Abi > __x)
- template<typename _L1, typename _L2, typename... _L3>
  int try_lock (_L1 &__l1, _L2 &__l2, _L3 &... __l3)
- template<typename... _UTypes>
  tuple (_UTypes...) -> tuple< _UTypes... >
- template<typename _Alloc, typename... _UTypes>
  tuple (allocator_arg_t, _Alloc, _UTypes...) -> tuple< _UTypes... >
- template<typename _Alloc, typename _T1, typename _T2>
  tuple (allocator_arg_t, _Alloc, pair< _T1, _T2 >) -> tuple< _T1, _T2 >
- template<typename _Alloc, typename... _UTypes>
  tuple (allocator_arg_t, _Alloc, tuple< _UTypes... >) -> tuple< _UTypes... >
- template<typename _T1, typename _T2>
  tuple (pair< _T1, _T2 >) -> tuple< _T1, _T2 >

- template<typename... _Tpls, typename = typename enable_if<__and_<__is_tuple_like<_Tpls>...>::value>::type>
  constexpr auto tuple_cat (_Tpls &&... __tpls) -> typename __tuple_cat_result< _Tpls... >::__type
- bool uncaught_exception () noexcept
- void undeclare_no_pointers (char *, size_t)
- template<typename _Tp>
  _Tp * undeclare_reachable (_Tp *__p)
- void unexpected ()
- template<typename _Tp, typename _Alloc, typename... _Args>
  constexpr _Tp * **uninitialized_construct_using_allocator** (_Tp *__p, const _Alloc &__a, _Args &&... __args)
- template<typename _InputIterator, typename _ForwardIterator>
  _ForwardIterator uninitialized_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __result)
- template<typename _InputIterator, typename _Size, typename _ForwardIterator>
  _ForwardIterator uninitialized_copy_n (_InputIterator __first, _Size __n, _ForwardIterator __result)
- template<typename _ForwardIterator>
  void uninitialized_default_construct (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _Size>
  _ForwardIterator uninitialized_default_construct_n (_ForwardIterator __first, _Size __count)
- template<typename _ForwardIterator, typename _Tp>
  void uninitialized_fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__x)
- template<typename _ForwardIterator, typename _Size, typename _Tp>
  _ForwardIterator uninitialized_fill_n (_ForwardIterator __first, _Size __n, const _Tp &__x)
- template<typename _InputIterator, typename _ForwardIterator>
  _ForwardIterator uninitialized_move (_InputIterator __first, _InputIterator __last, _ForwardIterator __result)
- template<typename _InputIterator, typename _Size, typename _ForwardIterator>
  pair< _InputIterator, _ForwardIterator > uninitialized_move_n (_InputIterator __first, _Size __count, _Forward↩
  Iterator __result)
- template<typename _ForwardIterator>
  void uninitialized_value_construct (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _Size>
  _ForwardIterator uninitialized_value_construct_n (_ForwardIterator __first, _Size __count)
- template<typename _FIter>
  constexpr _FIter **unique** (_FIter, _FIter)
- template<typename _FIter, typename _BinaryPredicate>
  constexpr _FIter **unique** (_FIter, _FIter, _BinaryPredicate)
- template<typename _ForwardIterator>
  constexpr _ForwardIterator unique (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _BinaryPredicate>
  constexpr _ForwardIterator unique (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __↩
  binary_pred)
- template<typename _IIter, typename _OIter>
  constexpr _OIter **unique_copy** (_IIter, _IIter, _OIter)
- template<typename _IIter, typename _OIter, typename _BinaryPredicate>
  constexpr _OIter **unique_copy** (_IIter, _IIter, _OIter, _BinaryPredicate)
- template<typename _InputIterator, typename _OutputIterator>
  constexpr _OutputIterator unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result)
- template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate>
  constexpr _OutputIterator unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _↩
  BinaryPredicate __binary_pred)
- ios_base & unitbuf (ios_base &__base)
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩
  Allocator<_Allocator>>
  unordered_map (_InputIterator, _InputIterator, _Allocator) -> unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator > >, equal_to< __iter_key_t< _InputIterator > >, _Allocator >

- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  unordered_map (_InputIterator, _InputIterator, typename unordered_map< int, int >::size_type, _Allocator) -> unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator > >, equal_to< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  unordered_map (_InputIterator, _InputIterator, typename unordered_map< int, int >::size_type, _Hash, ↩ _Allocator) -> unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Hash, equal_to< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Hash = hash<__iter_key_t<_InputIterator>>, typename _Pred = equal_to<__iter_key_↩ t<_InputIterator>>, typename _Allocator = allocator<__iter_to_alloc_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_↩ Allocator>>
  unordered_map (_InputIterator, _InputIterator, typename unordered_map< int, int >::size_type={}, _Hash=_↩ Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_map< __iter_key_t< _InputIterator >, __iter_↩ val_t< _InputIterator >, _Hash, _Pred, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  unordered_map (initializer_list< pair< _Key, _Tp > >, _Allocator) -> unordered_map< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  unordered_map (initializer_list< pair< _Key, _Tp > >, typename unordered_map< int, int >::size_type, _↩ Allocator) -> unordered_map< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  unordered_map (initializer_list< pair< _Key, _Tp > >, typename unordered_map< int, int >::size_type, _Hash, _Allocator) -> unordered_map< _Key, _Tp, _Hash, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  unordered_map (initializer_list< pair< _Key, _Tp > >, typename unordered_map< int, int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_map< _Key, _Tp, _Hash, _Pred, _↩ Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  unordered_multimap (_InputIterator, _InputIterator, _Allocator) -> unordered_multimap< __iter_key_t< _↩ InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator > >, equal_to< __iter_↩ key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  unordered_multimap (_InputIterator, _InputIterator, unordered_multimap< int, int >::size_type, _Allocator) -> unordered_multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator > >, equal_to< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = ↩ _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  unordered_multimap (_InputIterator, _InputIterator, unordered_multimap< int, int >::size_type, _Hash, _↩ Allocator) -> unordered_multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Hash, equal_to< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Hash = hash<__iter_key_t<_InputIterator>>, typename _Pred = equal_to<__iter_key_↩ t<_InputIterator>>, typename _Allocator = allocator<__iter_to_alloc_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_↩ Allocator>>
  unordered_multimap (_InputIterator, _InputIterator, unordered_multimap< int, int >::size_type={}, _Hash=_↩

Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Hash, _Pred, _Allocator >

- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  unordered_multimap (initializer_list< pair< _Key, _Tp > >, _Allocator) -> unordered_multimap< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  unordered_multimap (initializer_list< pair< _Key, _Tp > >, unordered_multimap< int, int >::size_type, _Allocator) -> unordered_multimap< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  unordered_multimap (initializer_list< pair< _Key, _Tp > >, unordered_multimap< int, int >::size_type, _Hash, _Allocator) -> unordered_multimap< _Key, _Tp, _Hash, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  unordered_multimap (initializer_list< pair< _Key, _Tp > >, unordered_multimap< int, int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_multimap< _Key, _Tp, _Hash, _Pred, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  unordered_multiset (_InputIterator, _InputIterator, _Allocator) -> unordered_multiset< typename iterator_traits< _InputIterator >::value_type, hash< typename iterator_traits< _InputIterator >::value_type >, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  unordered_multiset (_InputIterator, _InputIterator, unordered_multiset< int >::size_type, _Allocator) -> unordered_multiset< typename iterator_traits< _InputIterator >::value_type, hash< typename iterator_traits< _InputIterator >::value_type >, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  unordered_multiset (_InputIterator, _InputIterator, unordered_multiset< int >::size_type, _Hash, _Allocator) -> unordered_multiset< typename iterator_traits< _InputIterator >::value_type, _Hash, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Hash = hash<typename iterator_traits<_InputIterator>::value_type>, typename _Pred = equal_to<typename iterator_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  unordered_multiset (_InputIterator, _InputIterator, unordered_multiset< int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_multiset< typename iterator_traits< _InputIterator >::value_type, _Hash, _Pred, _Allocator >
- template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  unordered_multiset (initializer_list< _Tp >, _Allocator) -> unordered_multiset< _Tp, hash< _Tp >, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  unordered_multiset (initializer_list< _Tp >, unordered_multiset< int >::size_type, _Allocator) -> unordered_multiset< _Tp, hash< _Tp >, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  unordered_multiset (initializer_list< _Tp >, unordered_multiset< int >::size_type, _Hash, _Allocator) -> unordered_multiset< _Tp, _Hash, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Hash = hash<_Tp>, typename _Pred = equal_to<_Tp>, typename _Allocator = allocator<_Tp>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>

unordered_multiset (initializer_list< _Tp >, unordered_multiset< int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_multiset< _Tp, _Hash, _Pred, _Allocator >

- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  unordered_set (_InputIterator, _InputIterator, _Allocator) -> unordered_set< typename iterator_traits< _InputIterator >::value_type, hash< typename iterator_traits< _InputIterator >::value_type >, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  unordered_set (_InputIterator, _InputIterator, unordered_set< int >::size_type, _Allocator) -> unordered_set< typename iterator_traits< _InputIterator >::value_type, hash< typename iterator_traits< _InputIterator >::value_type >, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  unordered_set (_InputIterator, _InputIterator, unordered_set< int >::size_type, _Hash, _Allocator) -> unordered_set< typename iterator_traits< _InputIterator >::value_type, _Hash, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Hash = hash<typename iterator_traits<_InputIterator>::value_type>, typename _Pred = equal_to<typename iterator_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  unordered_set (_InputIterator, _InputIterator, unordered_set< int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_set< typename iterator_traits< _InputIterator >::value_type, _Hash, _Pred, _Allocator >
- template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  unordered_set (initializer_list< _Tp >, _Allocator) -> unordered_set< _Tp, hash< _Tp >, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  unordered_set (initializer_list< _Tp >, unordered_set< int >::size_type, _Allocator) -> unordered_set< _Tp, hash< _Tp >, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  unordered_set (initializer_list< _Tp >, unordered_set< int >::size_type, _Hash, _Allocator) -> unordered_set< _Tp, _Hash, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Hash = hash<_Tp>, typename _Pred = equal_to<_Tp>, typename _Allocator = allocator<_Tp>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  unordered_set (initializer_list< _Tp >, unordered_set< int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_set< _Tp, _Hash, _Pred, _Allocator >
- template<typename _FIter, typename _Tp>
  constexpr _FIter **upper_bound** (_FIter, _FIter, const _Tp &)
- template<typename _FIter, typename _Tp, typename _Compare>
  constexpr _FIter **upper_bound** (_FIter, _FIter, const _Tp &, _Compare)
- template<typename _ForwardIterator, typename _Tp>
  constexpr _ForwardIterator upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)
- template<typename _ForwardIterator, typename _Tp, typename _Compare>
  constexpr _ForwardIterator upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)
- ios_base & uppercase (ios_base &__base)
- template<typename _Facet>
  const _Facet & use_facet (const locale &__loc)
- template const codecvt< char, char, mbstate_t > & **use_facet**< **codecvt**< **char, char, mbstate_t** > > (const locale &)

- template const codecvt< wchar_t, char, mbstate_t > & **use_facet**< **codecvt**< **wchar_t, char, mbstate_t** > > (const locale &)
- template<_Std_pair _Tp, typename _Alloc>
  constexpr auto **uses_allocator_construction_args** (const _Alloc &) noexcept
- template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>
  constexpr auto **uses_allocator_construction_args** (const _Alloc &, _Up &&, _Vp &&) noexcept
- template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>
  constexpr auto **uses_allocator_construction_args** (const _Alloc &, const pair< _Up, _Vp > &) noexcept
- template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>
  constexpr auto **uses_allocator_construction_args** (const _Alloc &, pair< _Up, _Vp > &&) noexcept
- template<typename _Tp, typename _Alloc, typename... _Args>
  requires (! _Std_pair<_Tp>)
  constexpr auto **uses_allocator_construction_args** (const _Alloc &__a, _Args &&... __args) noexcept
- template<_Std_pair _Tp, typename _Alloc, typename _Tuple1, typename _Tuple2>
  constexpr auto **uses_allocator_construction_args** (const _Alloc &__a, piecewise_construct_t, _Tuple1 &&__x, _Tuple2 &&__y) noexcept
- template<typename _Tp, size_t _Nm>
  valarray (const _Tp(&)[_Nm], size_t) -> valarray< _Tp >
- template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  vector (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> vector< _ValT, _Allocator >
- wchar_t * **wcschr** (wchar_t *__p, wchar_t __c)
- wchar_t * **wcspbrk** (wchar_t *__s1, const wchar_t *__s2)
- wchar_t * **wcsrchr** (wchar_t *__p, wchar_t __c)
- wchar_t * **wcsstr** (wchar_t *__s1, const wchar_t *__s2)
- template<typename _Tp>
  weak_ptr (shared_ptr< _Tp >) -> weak_ptr< _Tp >
- wchar_t * **wmemchr** (wchar_t *__p, wchar_t __c, size_t __n)
- template<typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & ws (basic_istream< _CharT, _Traits > &__is)

- template<size_t _Nb>
  constexpr bitset< _Nb > operator& (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept
- template<size_t _Nb>
  constexpr bitset< _Nb > operator| (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept
- template<size_t _Nb>
  constexpr bitset< _Nb > operator^ (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept

- template<class _CharT, class _Traits, size_t _Nb>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, bitset< _Nb > &__x)
- template<class _CharT, class _Traits, size_t _Nb>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const bitset< _Nb > &__x)

- template<typename _Tp>
  constexpr complex< _Tp > operator+ (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp>
  constexpr complex< _Tp > operator+ (const complex< _Tp > &__x, const _Tp &__y)

- template<typename _Tp>
  constexpr complex< _Tp > operator+ (const _Tp &__x, const complex< _Tp > &__y)

- template<typename _Tp>
  constexpr complex< _Tp > operator- (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp>
  constexpr complex< _Tp > operator- (const complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp>
  constexpr complex< _Tp > operator- (const _Tp &__x, const complex< _Tp > &__y)

- template<typename _Tp>
  constexpr complex< _Tp > operator* (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp>
  constexpr complex< _Tp > operator* (const complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp>
  constexpr complex< _Tp > operator* (const _Tp &__x, const complex< _Tp > &__y)

- template<typename _Tp>
  constexpr complex< _Tp > operator/ (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp>
  constexpr complex< _Tp > operator/ (const complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp>
  constexpr complex< _Tp > operator/ (const _Tp &__x, const complex< _Tp > &__y)

- template<typename _Tp>
  constexpr bool operator== (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp>
  constexpr bool operator== (const complex< _Tp > &__x, const _Tp &__y)

- template<typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__in, _CharT &__c)
- template<class _Traits>
  basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits > &__in, unsigned char &__c)
- template<class _Traits>
  basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits > &__in, signed char &__c)

- template<typename _CharT, typename _Traits, size_t _Num>
  basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__in, _CharT(&__s)[_↩
  Num])
- template<class _Traits, size_t _Num>
  basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits > &__in, unsigned char(&__s)[_↩
  Num])
- template<class _Traits, size_t _Num>
  basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits > &__in, signed char(&__s)[_Num])

- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, _CharT __c)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, char __c)
- template<typename _Traits>
  basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, char __c)
- template<typename _Traits>
  basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, signed char __c)
- template<typename _Traits>
  basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, unsigned char __c)
- template<typename _Traits>
  basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &, wchar_t)=delete
- template<typename _Traits>
  basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &, char16_t)=delete
- template<typename _Traits>
  basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &, char32_t)=delete
- template<typename _Traits>
  basic_ostream< wchar_t, _Traits > & operator<< (basic_ostream< wchar_t, _Traits > &, char16_t)=delete
- template<typename _Traits>
  basic_ostream< wchar_t, _Traits > & operator<< (basic_ostream< wchar_t, _Traits > &, char32_t)=delete

- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, const _CharT *__s)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, const char *__s)
- template<typename _Traits>
  basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, const char *__s)
- template<typename _Traits>
  basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, const signed char *__s)
- template<typename _Traits>
  basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, const unsigned char *__s)
- template<typename _Traits>
  basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &, const wchar_t *)=delete
- template<typename _Traits>
  basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &, const char16_t *)=delete
- template<typename _Traits>
  basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &, const char32_t *)=delete
- template<typename _Traits>
  basic_ostream< wchar_t, _Traits > & operator<< (basic_ostream< wchar_t, _Traits > &, const char16_t *)=delete
- template<typename _Traits>
  basic_ostream< wchar_t, _Traits > & operator<< (basic_ostream< wchar_t, _Traits > &, const char32_t *)=delete

Matching, Searching, and Replacing

- template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits>
  bool regex_match (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)

- template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>
  bool regex_match (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Ch_type, typename _Alloc, typename _Rx_traits>
  bool regex_match (const _Ch_type *__s, match_results< const _Ch_type *, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_constants::match_default)
- template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>
  bool regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex< _Ch↩ _type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>
  bool regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &&, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &, const basic_regex< _Ch_type, _Rx_traits > &, regex_constants::match_flag_type=regex_constants::match_default)=delete
- template<typename _Ch_type, class _Rx_traits>
  bool regex_match (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_constants::match_default)
- template<typename _Ch_traits, typename _Str_allocator, typename _Ch_type, typename _Rx_traits>
  bool regex_match (const basic_string< _Ch_type, _Ch_traits, _Str_allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits>
  bool regex_search (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>
  bool regex_search (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Ch_type, class _Alloc, class _Rx_traits>
  bool regex_search (const _Ch_type *__s, match_results< const _Ch_type *, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)
- template<typename _Ch_type, typename _Rx_traits>
  bool regex_search (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)
- template<typename _Ch_traits, typename _String_allocator, typename _Ch_type, typename _Rx_traits>
  bool regex_search (const basic_string< _Ch_type, _Ch_traits, _String_allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>
  bool regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex< _Ch↩ _type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)
- template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>
  bool regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &&, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &, const basic_regex< _Ch_type, _Rx_traits > &, regex_constants::match_flag_type=regex_constants::match_default)=delete
- template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa>
  _Out_iter regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, ↩ _Rx_traits > &__e, const basic_string< _Ch_type, _St, _Sa > &__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type>
  _Out_iter regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _↩ Rx_traits > &__e, const _Ch_type *__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa, typename _Fst, typename _Fsa>
  basic_string< _Ch_type, _St, _Sa > regex_replace (const basic_string< _Ch_type, _St, _Sa > &__↩ s, const basic_regex< _Ch_type, _Rx_traits > &__e, const basic_string< _Ch_type, _Fst, _Fsa > &__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa>
  basic_string< _Ch_type, _St, _Sa > regex_replace (const basic_string< _Ch_type, _St, _Sa > &__s, const

basic_regex< _Ch_type, _Rx_traits > &__e, const _Ch_type *__fmt, regex_constants::match_flag_type __↩ flags=regex_constants::match_default)

- template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa>
  basic_string< _Ch_type > regex_replace (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_↩ traits > &__e, const basic_string< _Ch_type, _St, _Sa > &__fmt, regex_constants::match_flag_type __↩ flags=regex_constants::match_default)
- template<typename _Rx_traits, typename _Ch_type>
  basic_string< _Ch_type > regex_replace (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits > &__e, const _Ch_type *__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)

- template<typename _Tp, _Lock_policy _Lp>
  bool atomic_is_lock_free (const __shared_ptr< _Tp, _Lp > *)
- template<typename _Tp>
  bool atomic_is_lock_free (const shared_ptr< _Tp > *__p)

- template<typename _Tp>
  shared_ptr< _Tp > atomic_load_explicit (const shared_ptr< _Tp > *__p, memory_order)
- template<typename _Tp>
  shared_ptr< _Tp > atomic_load (const shared_ptr< _Tp > *__p)
- template<typename _Tp, _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > atomic_load_explicit (const __shared_ptr< _Tp, _Lp > *__p, memory_order)
- template<typename _Tp, _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > atomic_load (const __shared_ptr< _Tp, _Lp > *__p)

- template<typename _Tp>
  void atomic_store_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r, memory_order)
- template<typename _Tp>
  void atomic_store (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r)
- template<typename _Tp, _Lock_policy _Lp>
  void atomic_store_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r, memory_order)
- template<typename _Tp, _Lock_policy _Lp>
  void atomic_store (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r)

- template<typename _Tp>
  shared_ptr< _Tp > atomic_exchange_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __↩ r, memory_order)
- template<typename _Tp>
  shared_ptr< _Tp > atomic_exchange (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r)
- template<typename _Tp, _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > atomic_exchange_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r, memory_order)
- template<typename _Tp, _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > atomic_exchange (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r)

- template<typename _Tp>
  bool atomic_compare_exchange_strong_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__↩ v, shared_ptr< _Tp > __w, memory_order, memory_order)

- template<typename _Tp>
  bool atomic_compare_exchange_strong (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w)
- template<typename _Tp>
  bool atomic_compare_exchange_weak_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w, memory_order __success, memory_order __failure)
- template<typename _Tp>
  bool atomic_compare_exchange_weak (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w)
- template<typename _Tp, _Lock_policy _Lp>
  bool atomic_compare_exchange_strong_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w, memory_order, memory_order)
- template<typename _Tp, _Lock_policy _Lp>
  bool atomic_compare_exchange_strong (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w)
- template<typename _Tp, _Lock_policy _Lp>
  bool atomic_compare_exchange_weak_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w, memory_order __success, memory_order __failure)
- template<typename _Tp, _Lock_policy _Lp>
  bool atomic_compare_exchange_weak (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, _shared_ptr< _Tp, _Lp > __w)

- template<size_t _Int, class _Tp1, class _Tp2>
  constexpr tuple_element< _Int, pair< _Tp1, _Tp2 > >::type & get (pair< _Tp1, _Tp2 > &__in) noexcept
- template<size_t _Int, class _Tp1, class _Tp2>
  constexpr tuple_element< _Int, pair< _Tp1, _Tp2 > >::type && get (pair< _Tp1, _Tp2 > &&__in) noexcept
- template<size_t _Int, class _Tp1, class _Tp2>
  constexpr const tuple_element< _Int, pair< _Tp1, _Tp2 > >::type & get (const pair< _Tp1, _Tp2 > &__in) noexcept
- template<size_t _Int, class _Tp1, class _Tp2>
  constexpr const tuple_element< _Int, pair< _Tp1, _Tp2 > >::type && get (const pair< _Tp1, _Tp2 > &&__in) noexcept
- template<typename _Tp, typename _Up>
  constexpr _Tp & get (pair< _Tp, _Up > &__p) noexcept
- template<typename _Tp, typename _Up>
  constexpr const _Tp & get (const pair< _Tp, _Up > &__p) noexcept
- template<typename _Tp, typename _Up>
  constexpr _Tp && get (pair< _Tp, _Up > &&__p) noexcept
- template<typename _Tp, typename _Up>
  constexpr const _Tp && get (const pair< _Tp, _Up > &&__p) noexcept
- template<typename _Tp, typename _Up>
  constexpr _Tp & get (pair< _Up, _Tp > &__p) noexcept
- template<typename _Tp, typename _Up>
  constexpr const _Tp & get (const pair< _Up, _Tp > &__p) noexcept
- template<typename _Tp, typename _Up>
  constexpr _Tp && get (pair< _Up, _Tp > &&__p) noexcept
- template<typename _Tp, typename _Up>
  constexpr const _Tp && get (const pair< _Up, _Tp > &&__p) noexcept

Variables

- template<typename _ValT, typename _Tp>
  constexpr bool **__can_use_memchr_for_find**
- const _Lock_policy **__default_lock_policy**
- template<typename _Tp>
  constexpr auto **__denorm_min_v**
- template<typename _Tp>
  constexpr auto **__digits10_v**
- template<typename _Tp>
  constexpr auto **__digits_v**
- template<typename _Tp>
  constexpr auto **__epsilon_v**
- template<typename _Tp>
  constexpr auto **__finite_max_v**
- template<typename _Tp>
  constexpr auto **__finite_min_v**
- template<template< typename > class _Trait, typename _Tp>
  constexpr bool **__has_iec559_behavior_v**
- template<typename _Tp>
  constexpr bool **__has_iec559_storage_format_v**
- template<typename _Tp>
  constexpr auto **__infinity_v**
- static ios_base::Init **__ioinit**
- template<typename _Tp, typename = void>
  constexpr bool **__is_hash_enabled_for**
- template<typename _Tp>
  constexpr bool **__is_hash_enabled_for**< **_Tp, __void_t**< **decltype(hash**< **_Tp** >()(**declval**< **_Tp** >()))**> >
- template<typename>
  constexpr bool **__is_in_place_index_v**
- template<size_t _Nm>
  constexpr bool **__is_in_place_index_v**< **in_place_index_t**< **_Nm** > >
- template<typename>
  constexpr bool **__is_in_place_type_v**
- template<typename _Tp>
  constexpr bool **__is_in_place_type_v**< **in_place_type_t**< **_Tp** > >
- template<typename _Tp, typename... _Args>
  constexpr bool **__is_nothrow_new_constructible**
- template<typename _Tp>
  constexpr bool **__is_pair**
- template<typename _Tp, typename _Up>
  constexpr bool **__is_pair**< **pair**< **_Tp, _Up** > >
- template<typename _Tp>
  constexpr auto **__max_digits10_v**
- template<typename _Tp>
  constexpr auto **__max_exponent10_v**
- template<typename _Tp>
  constexpr auto **__max_exponent_v**
- template<typename _Tp>
  constexpr auto **__min_exponent10_v**
- template<typename _Tp>
  constexpr auto **__min_exponent_v**

- template<typename _Tp>
  constexpr auto **__norm_min_v**
- template<typename _Tp>
  constexpr auto **__quiet_NaN_v**
- template<typename _Tp>
  constexpr auto **__radix_v**
- template<typename _Tp>
  constexpr auto **__reciprocal_overflow_threshold_v**
- template<typename _Tp>
  constexpr auto **__round_error_v**
- template<typename _Tp>
  constexpr auto **__signaling_NaN_v**
- template<typename _CharT>
  locale::id **__timepunct**< **_CharT** >**::id**
- template<typename _CharT>
  const _CharT * **__timepunct_cache**< **_CharT** >**::_S_timezones** [14]
- template<> const char * **__timepunct_cache**< **char** >**::_S_timezones** [14]
- template<> const wchar_t * **__timepunct_cache**< **wchar_t** >**::_S_timezones** [14]
- template<template< typename > class _Trait, typename _Tp>
  constexpr bool **__value_exists_v**
- constexpr adopt_lock_t adopt_lock
- template<size_t _Len, typename... _Types>
  const size_t **aligned_union**< **_Len, _Types...** >**::alignment_value**
- template<typename _CharT, typename _Traits, typename _Alloc>
  const basic_string< _CharT, _Traits, _Alloc >::size_type **basic_string**< **_CharT, _Traits, _Alloc** >**::npos**
- template<typename _InternT, typename _ExternT, typename _StateT>
  locale::id **codecvt**< **_InternT, _ExternT, _StateT** >**::id**
- template<typename _InternT, typename _ExternT>
  locale::id **codecvt**< **_InternT, _ExternT, encoding_state** >**::id**
- template<typename _CharT>
  locale::id **collate**< **_CharT** >**::id**
- constexpr __compare::_Partial_fallback **compare_partial_order_fallback**
- constexpr __compare::_Strong_fallback **compare_strong_order_fallback**
- constexpr __compare::_Weak_fallback **compare_weak_order_fallback**
- template<typename _CharT>
  locale::id **ctype**< **_CharT** >**::id**
- constexpr default_sentinel_t default_sentinel
- constexpr defer_lock_t defer_lock
- constexpr destroying_delete_t destroying_delete
- template<typename _Iterator1, typename _Iterator2>
  constexpr bool **disable_sized_sentinel_for**< **move_iterator**< **_Iterator1** >, **move_iterator**< **_Iterator2** > >
- template<typename _Iterator1, typename _Iterator2>
  constexpr bool **disable_sized_sentinel_for**< **reverse_iterator**< **_Iterator1** >, **reverse_iterator**< **_Iterator2** > >
- constexpr _Swallow_assign ignore
- constexpr in_place_t **in_place**
- template<size_t _Idx>
  constexpr in_place_index_t< _Idx > **in_place_index**
- template<typename _Tp>
  constexpr in_place_type_t< _Tp > **in_place_type**
- template<typename _Tp>
  constexpr bool **is_bind_expression_v**

- template<typename _Tp>
  constexpr bool **is_error_code_enum_v**
- template<typename _Tp>
  constexpr bool **is_error_condition_enum_v**
- template<typename _Tp>
  constexpr int **is_placeholder_v**
- constexpr memory_order **memory_order_acq_rel**
- constexpr memory_order **memory_order_acquire**
- constexpr memory_order **memory_order_consume**
- constexpr memory_order **memory_order_relaxed**
- constexpr memory_order **memory_order_release**
- constexpr memory_order **memory_order_seq_cst**
- template<typename _CharT>
  locale::id **messages< _CharT >::id**
- template<typename _CharT, typename _InIter>
  locale::id **money_get< _CharT, _InIter >::id**
- template<typename _CharT, typename _OutIter>
  locale::id **money_put< _CharT, _OutIter >::id**
- template<typename _CharT, bool _Intl>
  locale::id **moneypunct< _CharT, _Intl >::id**
- template<typename _CharT, bool _Intl>
  const bool **moneypunct< _CharT, _Intl >::intl**
- template<typename _CharT, bool _Intl>
  const bool **moneypunct_byname< _CharT, _Intl >::intl**
- const nothrow_t **nothrow**
- template<typename _CharT, typename _InIter>
  locale::id **num_get< _CharT, _InIter >::id**
- template<typename _CharT, typename _OutIter>
  locale::id **num_put< _CharT, _OutIter >::id**
- template<typename _CharT>
  locale::id **numpunct< _CharT >::id**
- constexpr __compare::_Partial_order **partial_order**
- constexpr piecewise_construct_t piecewise_construct
- template<typename _R1, typename _R2>
  constexpr bool **ratio_equal_v**
- template<typename _R1, typename _R2>
  constexpr bool **ratio_greater_equal_v**
- template<typename _R1, typename _R2>
  constexpr bool **ratio_greater_v**
- template<typename _R1, typename _R2>
  constexpr bool **ratio_less_equal_v**
- template<typename _R1, typename _R2>
  constexpr bool **ratio_less_v**
- template<typename _R1, typename _R2>
  constexpr bool **ratio_not_equal_v**
- constexpr __compare::_Strong_order **strong_order**
- template<typename _CharT, typename _InIter>
  locale::id **time_get< _CharT, _InIter >::id**
- template<typename _CharT, typename _OutIter>
  locale::id **time_put< _CharT, _OutIter >::id**
- constexpr try_to_lock_t try_to_lock

- template<typename _Tp>
  constexpr size_t **tuple_size_v**
- template<typename _Tp, size_t _Nm>
  constexpr size_t **tuple_size_v**< **array**< **_Tp, _Nm** > >
- template<typename _Tp, size_t _Nm>
  constexpr size_t **tuple_size_v**< **const array**< **_Tp, _Nm** > >
- template<typename _Tp1, typename _Tp2>
  constexpr size_t **tuple_size_v**< **const pair**< **_Tp1, _Tp2** > >
- template<typename... _Types>
  constexpr size_t **tuple_size_v**< **const tuple**< **_Types...** > >
- template<typename _Tp1, typename _Tp2>
  constexpr size_t **tuple_size_v**< **pair**< **_Tp1, _Tp2** > >
- template<typename... _Types>
  constexpr size_t **tuple_size_v**< **tuple**< **_Types...** > >
- constexpr __compare::_Weak_order **weak_order**

Standard Stream Objects

The <iostream> header declares the eight standard stream objects. For other declarations, see https://gcc.gnu.org/onlinedocs/libstdc++/manual/io.html and the I/O forward declarations

They are required by default to cooperate with the global C library's FILE streams, and to be available during program startup and termination. For more information, see the section of the manual linked to above.

- istream cin
- ostream cout
- ostream cerr
- ostream clog
- wistream wcin
- wostream wcout
- wostream wcerr
- wostream wclog

- template<typename... _Tp>
  using common_reference_t
- template<typename _Tp, typename _Up>
  constexpr bool is_layout_compatible_v
- template<typename _Base, typename _Derived>
  constexpr bool is_pointer_interconvertible_base_of_v
- template<typename _S1, typename _S2, typename _M1, typename _M2>
  constexpr bool is_corresponding_member (_M1 _S1::*__m1, _M2 _S2::*__m2) noexcept
- template<typename _Tp, typename _Mem>
  constexpr bool is_pointer_interconvertible_with_class (_Mem _Tp::*__mp) noexcept

4.10.1 Detailed Description

ISO C++ entities toplevel namespace is std.

4.10.2 Typedef Documentation

__ptr_rebind

```
template<typename _Ptr, typename _Tp>
using std::__ptr_rebind
```

Convenience alias for rebinding pointers.

__umap_traits

```
template<bool _Cache>
using std::__umap_traits
```
Base types for unordered_map.

__ummap_traits

```
template<bool _Cache>
using std::__ummap_traits
```
Base types for unordered_multimap.

__umset_traits

```
template<bool _Cache>
using std::__umset_traits
```
Base types for unordered_multiset.

__uset_traits

```
template<bool _Cache>
using std::__uset_traits
```
Base types for unordered_set.

compare_three_way_result_t

```
template<typename _Tp, typename _Up = _Tp>
using std::compare_three_way_result_t
```
[cmp.result], result of three-way comparison

index_sequence

```
template<size_t... _Idx>
using std::index_sequence
```
Alias template index_sequence.

index_sequence_for

```
template<typename... _Types>
using std::index_sequence_for
```
Alias template index_sequence_for.

make_index_sequence

```
template<size_t _Num>
using std::make_index_sequence
```
Alias template make_index_sequence.

make_integer_sequence

```
template<typename _Tp, _Tp _Num>
using std::make_integer_sequence
```
Alias template make_integer_sequence.

new_handler

```
typedef void(* std::new_handler) ()
```
If you write your own error handler to be called by `new`, it must be of this type.

streamoff

```
typedef long long std::streamoff
```
Type used by fpos, char_traits<char>, and char_traits<wchar_t>.
In clauses 21.1.3.1 and 27.4.1 streamoff is described as an implementation defined type. Note: In versions of GCC up to and including GCC 3.3, streamoff was typedef long.

streampos

```
typedef fpos<mbstate_t> std::streampos
```
File position for char streams.

streamsize

```
typedef ptrdiff_t std::streamsize
```
Integral type for I/O operation counts and buffer sizes.

u16streampos

```
typedef fpos<mbstate_t> std::u16streampos
```
File position for char16_t streams.

u32streampos

```
typedef fpos<mbstate_t> std::u32streampos
```
File position for char32_t streams.

wstreampos

```
typedef fpos<mbstate_t> std::wstreampos
```
File position for wchar_t streams.

4.10.3 Enumeration Type Documentation

chars_format

```
enum class std::chars_format  [strong]
```
floating-point format for primitive numerical conversion

float_denorm_style

```
enum std::float_denorm_style
```
Describes the denormalization for floating-point types.
These values represent the presence or absence of a variable number of exponent bits. This type is used in the std↩ ::numeric_limits class.

Enumerator

denorm_indeterminate	Indeterminate at compile time whether denormalized values are allowed.
denorm_absent	The type does not allow denormalized values.
denorm_present	The type allows denormalized values.

float_round_style

```
enum std::float_round_style
```
Describes the rounding style for floating-point types.
This is used in the std::numeric_limits class.

Enumerator

round_indeterminate	Intermediate.
round_toward_zero	To zero.
round_to_nearest	To the nearest representable value.
round_toward_infinity	To infinity.
round_toward_neg_infinity	To negative infinity.

io_errc

```
enum class std::io_errc  [strong]
```
I/O error code.

4.10.4 Function Documentation

__find_if_not()

```
template<typename _InputIterator, typename _Predicate>
_InputIterator std::__find_if_not (
             _InputIterator __first,
             _InputIterator __last,
             _Predicate __pred)  [inline], [constexpr]
```
Provided for stable_partition to use.
Referenced by find_if_not().

__find_if_not_n()

```
template<typename _InputIterator, typename _Predicate, typename _Distance>
_InputIterator std::__find_if_not_n (
             _InputIterator __first,
             _Distance & __len,
             _Predicate __pred)  [constexpr]
```
Like find_if_not(), but uses and updates a count of the remaining range length instead of comparing against an end iterator.
Referenced by __stable_partition_adaptive().

__gcd()

```
template<typename _EuclideanRingElement>
_EuclideanRingElement std::__gcd (
             _EuclideanRingElement __m,
             _EuclideanRingElement __n)  [constexpr]
```
This is a helper function for the rotate algorithm specialized on RAIs. It returns the greatest common divisor of two integer values.

__gen_two_uniform_ints()

```
template<typename _IntType, typename _UniformRandomBitGenerator>
pair< _IntType, _IntType > std::__gen_two_uniform_ints (
            _IntType __b0,
            _IntType __b1,
            _UniformRandomBitGenerator && __g)
```

Generate two uniformly distributed integers using a single distribution invocation.

Parameters

__b0	The upper bound for the first integer.
__b1	The upper bound for the second integer.
__g	A UniformRandomBitGenerator.

Returns

A pair (i, j) with i and j uniformly distributed over [0, __b0) and [0, __b1), respectively.

Requires: __b0 * __b1 <= __g.max() - __g.min().
Using uniform_int_distribution with a range that is very small relative to the range of the generator ends up wasting potentially expensively generated randomness, since uniform_int_distribution does not store leftover randomness between invocations.
If we know we want two integers in ranges that are sufficiently small, we can compose the ranges, use a single distribution invocation, and significantly reduce the waste.
References make_pair().
Referenced by __sample(), and shuffle().

__inplace_stable_sort()

```
template<typename _RandomAccessIterator, typename _Compare>
void std::__inplace_stable_sort (
            _RandomAccessIterator __first,
            _RandomAccessIterator __last,
            _Compare __comp)
```

This is a helper function for the stable sorting routines.
References __inplace_stable_sort(), and __merge_without_buffer().
Referenced by __inplace_stable_sort().

__lg()

```
template<typename _Tp>
_Tp std::__lg (
            _Tp __n)  [inline], [constexpr]
```

This is a helper function for the sort routines and for random.tcc.
Referenced by nth_element(), nth_element(), std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::operator()(), and std::linear_congruential_engine< uint_fast32_t, 16807UL, 0UL, 2147483647UL >::seed().

__merge_adaptive()

```
template<typename _BidirectionalIterator, typename _Distance, typename _Pointer, typename _↩
Compare>
void std::__merge_adaptive (
            _BidirectionalIterator __first,
            _BidirectionalIterator __middle,
```

```
            _BidirectionalIterator __last,
            _Distance __len1,
            _Distance __len2,
            _Pointer __buffer,
            _Compare __comp)
```

This is a helper function for the merge routines.
References __move_merge_adaptive(), and __move_merge_adaptive_backward().

__merge_without_buffer()

```
template<typename _BidirectionalIterator, typename _Distance, typename _Compare>
void std::__merge_without_buffer (
            _BidirectionalIterator __first,
            _BidirectionalIterator __middle,
            _BidirectionalIterator __last,
            _Distance __len1,
            _Distance __len2,
            _Compare __comp)
```

This is a helper function for the merge routines.
References __merge_without_buffer(), advance(), and distance().
Referenced by __inplace_stable_sort(), and __merge_without_buffer().

__move_median_to_first()

```
template<typename _Iterator, typename _Compare>
void std::__move_median_to_first (
            _Iterator __result,
            _Iterator __a,
            _Iterator __b,
            _Iterator __c,
            _Compare __comp)  [constexpr]
```

Swaps the median value of *__a, *__b and *__c under __comp to *__result.

__move_merge()

```
template<typename _InputIterator, typename _OutputIterator, typename _Compare>
_OutputIterator std::__move_merge (
            _InputIterator __first1,
            _InputIterator __last1,
            _InputIterator __first2,
            _InputIterator __last2,
            _OutputIterator __result,
            _Compare __comp)
```

This is a helper function for the __merge_sort_loop routines.

__move_merge_adaptive()

```
template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename ↩
_Compare>
void std::__move_merge_adaptive (
            _InputIterator1 __first1,
            _InputIterator1 __last1,
            _InputIterator2 __first2,
            _InputIterator2 __last2,
```

```
            _OutputIterator __result,
            _Compare __comp)
```

This is a helper function for the __merge_adaptive routines.
Referenced by __merge_adaptive().

__move_merge_adaptive_backward()

```
template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _Bidirectional↩
Iterator3, typename _Compare>
void std::__move_merge_adaptive_backward (
            _BidirectionalIterator1 __first1,
            _BidirectionalIterator1 __last1,
            _BidirectionalIterator2 __first2,
            _BidirectionalIterator2 __last2,
            _BidirectionalIterator3 __result,
            _Compare __comp)
```

This is a helper function for the __merge_adaptive routines.
Referenced by __merge_adaptive().

__partition() [1/2]

```
template<typename _BidirectionalIterator, typename _Predicate>
_BidirectionalIterator std::__partition (
            _BidirectionalIterator __first,
            _BidirectionalIterator __last,
            _Predicate __pred,
            bidirectional_iterator_tag )  [constexpr]
```

This is a helper function...

__partition() [2/2]

```
template<typename _ForwardIterator, typename _Predicate>
_ForwardIterator std::__partition (
            _ForwardIterator __first,
            _ForwardIterator __last,
            _Predicate __pred,
            forward_iterator_tag )  [constexpr]
```

This is a helper function...
Referenced by partition().

__reverse() [1/2]

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

This is an uglified reverse(_BidirectionalIterator, _BidirectionalIterator) overloaded for bidirectional iterators.
Referenced by __rotate(), and reverse().

__reverse() [2/2]

```
template<typename _RandomAccessIterator>
void std::__reverse (
            _RandomAccessIterator __first,
```

```
            _RandomAccessIterator __last,
            random_access_iterator_tag )  [constexpr]
```

This is an uglified reverse(_BidirectionalIterator, _BidirectionalIterator) overloaded for random access iterators.

__rotate() [1/3]

```
template<typename _BidirectionalIterator>
_BidirectionalIterator std::__rotate (
            _BidirectionalIterator __first,
            _BidirectionalIterator __middle,
            _BidirectionalIterator __last,
            bidirectional_iterator_tag )  [constexpr]
```

This is a helper function for the rotate algorithm.
References __reverse().

__rotate() [2/3]

```
template<typename _ForwardIterator>
_ForwardIterator std::__rotate (
            _ForwardIterator __first,
            _ForwardIterator __middle,
            _ForwardIterator __last,
            forward_iterator_tag )  [constexpr]
```

This is a helper function for the rotate algorithm.
Referenced by rotate().

__rotate() [3/3]

```
template<typename _RandomAccessIterator>
_RandomAccessIterator std::__rotate (
            _RandomAccessIterator __first,
            _RandomAccessIterator __middle,
            _RandomAccessIterator __last,
            random_access_iterator_tag )  [constexpr]
```

This is a helper function for the rotate algorithm.

__rotate_adaptive()

```
template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _Distance>
_BidirectionalIterator1 std::__rotate_adaptive (
            _BidirectionalIterator1 __first,
            _BidirectionalIterator1 __middle,
            _BidirectionalIterator1 __last,
            _Distance __len1,
            _Distance __len2,
            _BidirectionalIterator2 __buffer,
            _Distance __buffer_size)
```

This is a helper function for the merge routines.

__sample() [1/2]

```
template<typename _ForwardIterator, typename _OutputIterator, typename _Cat, typename _Size,
typename _UniformRandomBitGenerator>
_OutputIterator std::__sample (
            _ForwardIterator __first,
```

```
            _ForwardIterator __last,
            forward_iterator_tag ,
            _OutputIterator __out,
            _Cat ,
            _Size __n,
            _UniformRandomBitGenerator && __g)
```

Selection sampling algorithm.
References __gen_two_uniform_ints(), distance(), std::pair< _T1, _T2 >::first, min(), and std::pair< _T1, _T2 >::second.

__sample() [2/2]

```
template<typename _InputIterator, typename _RandomAccessIterator, typename _Size, typename _↩
UniformRandomBitGenerator>
_RandomAccessIterator std::__sample (
            _InputIterator __first,
            _InputIterator __last,
            input_iterator_tag ,
            _RandomAccessIterator __out,
            random_access_iterator_tag ,
            _Size __n,
            _UniformRandomBitGenerator && __g)
```

Reservoir sampling algorithm.

__search_n_aux() [1/2]

```
template<typename _ForwardIterator, typename _Integer, typename _UnaryPredicate>
_ForwardIterator std::__search_n_aux (
            _ForwardIterator __first,
            _ForwardIterator __last,
            _Integer __count,
            _UnaryPredicate __unary_pred,
            std::forward_iterator_tag )  [constexpr]
```

This is an helper function for search_n overloaded for forward iterators.

__search_n_aux() [2/2]

```
template<typename _RandomAccessIter, typename _Integer, typename _UnaryPredicate>
_RandomAccessIter std::__search_n_aux (
            _RandomAccessIter __first,
            _RandomAccessIter __last,
            _Integer __count,
            _UnaryPredicate __unary_pred,
            std::random_access_iterator_tag )  [constexpr]
```

This is an helper function for search_n overloaded for random access iterators.

__stable_partition_adaptive()

```
template<typename _ForwardIterator, typename _Pointer, typename _Predicate, typename _Distance>
_ForwardIterator std::__stable_partition_adaptive (
            _ForwardIterator __first,
            _ForwardIterator __last,
            _Predicate __pred,
            _Distance __len,
            _Pointer __buffer,
            _Distance __buffer_size)
```

This is a helper function... Requires __first != __last and !__pred(*__first) and __len == distance(__first, __last).
!__pred(*__first) allows us to guarantee that we don't move-assign an element onto itself.
References __find_if_not_n(), __stable_partition_adaptive(), and advance().
Referenced by __stable_partition_adaptive().

_Construct()

```
template<typename _Tp, typename...  _Args>
void std::_Construct (
            _Tp * __p,
            _Args &&...  __args)  [inline], [constexpr]
```

Constructs an object in existing memory by invoking an allocated object's constructor with an initializer.
References forward().
Referenced by std::allocator_traits< _OuterAlloc >::construct(), std::allocator_traits< allocator< _Tp > >::construct(), and std::allocator_traits< allocator< void > >::construct().

_Destroy() [1/2]

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

Destroy a range of objects. If the value_type of the object has a trivial destructor, the compiler should optimize all of this away, otherwise the objects' destructors must be invoked.
References _Destroy(), and addressof().
Referenced by std::vector< _Tp, polymorphic_allocator< _Tp > >::~vector(), _Destroy(), _Destroy_n(), std::deque< _Tp, _Alloc >::_M_
std::deque< _Tp, _Alloc >::_M_range_initialize(), std::allocator_traits< _OuterAlloc >::destroy(), std::allocator_traits< allocator< void >
std::vector< _Tp, _Alloc >::operator=(), and std::vector< _Tp, _Alloc >::reserve().

_Destroy() [2/2]

```
template<typename _Tp>
void std::_Destroy (
            _Tp * __pointer)  [inline], [constexpr]
```

Destroy the object pointed to by a pointer type.

_Destroy_n()

```
template<typename _ForwardIterator, typename _Size>
_ForwardIterator std::_Destroy_n (
            _ForwardIterator __first,
            _Size __count)  [inline], [constexpr]
```

Destroy a range of objects. If the value_type of the object has a trivial destructor, the compiler should optimize all of this away, otherwise the objects' destructors must be invoked.
References _Destroy(), addressof(), and advance().

acosh()

```
float std::acosh (
            float __x)  [constexpr]
```

Additional overloads.

advance()

```
template<typename _InputIterator, typename _Distance>
```

```
void std::advance (
            _InputIterator & __i,
            _Distance __n)  [inline], [constexpr]
```

A generalization of pointer arithmetic.

Parameters

_↩ _i	An input iterator.
_↩ _n	The *delta* by which to change __i.

Returns

Nothing.

This increments `i` by `n`. For bidirectional and random access iterators, __n may be negative, in which case __i is decremented.
For random access iterators, this uses their + and − operations and are constant time. For other iterator classes they are linear time.
References __iterator_category().
Referenced by __merge_without_buffer(), __stable_partition_adaptive(), _Destroy_n(), std::deque< _Tp, _Alloc >::_M_range_initialize(), __gnu_pbds::detail::pat_trie_base::_Node_citer< node, leaf, head, inode, const_iterator, iterator, _Alloc >::get_child(), __gnu_pbds::detail::pat_trie_base::_Node_iter< node, leaf, head, inode, const_iterator, iterator, _Alloc >::get_child(), and partition_point().

begin() [1/4]

```
template<typename _Container>
auto std::begin (
            _Container & __cont) -> decltype(__cont.begin())   [inline], [constexpr], [noexcept]
```

Return an iterator pointing to the first element of the container.

Parameters

__cont	Container.

begin() [2/4]

```
template<typename _Tp, size_t _Nm>
_Tp * std::begin (
            _Tp(&) __arr[_Nm])  [inline], [constexpr], [noexcept]
```

Return an iterator pointing to the first element of the array.

Parameters

__arr	Array.

begin() [3/4]

```
template<typename _Container>
auto std::begin (
            const _Container & __cont) -> decltype(__cont.begin())   [inline], [constexpr], [noexcept]
```

Return an iterator pointing to the first element of the const container.

Parameters

__cont	Container.

boolalpha()

```
ios_base & std::boolalpha (
            ios_base & __base)  [inline]
```

Calls base.setf(ios_base::boolalpha).
References std::ios_base::boolalpha.

cbegin()

```
template<typename _Container>
auto std::cbegin (
            const _Container & __cont) -> decltype(std::begin(__cont))   [constexpr], [noexcept]
```

Return an iterator pointing to the first element of the const container.

Parameters

__cont	Container.

References begin().
Referenced by std::vector< _Tp, _Alloc >::insert().

cend()

```
template<typename _Container>
auto std::cend (
            const _Container & __cont) -> decltype(std::end(__cont))   [constexpr], [noexcept]
```

Return an iterator pointing to one past the last element of the const container.

Parameters

__cont	Container.

References end().

const_pointer_cast()

```
template<typename _Tp, typename _Tp1, _Lock_policy _Lp>
__shared_ptr< _Tp, _Lp > std::const_pointer_cast (
            const __shared_ptr< _Tp1, _Lp > & __r)  [inline], [noexcept]
```

const_pointer_cast

crbegin()

```
template<typename _Container>
auto std::crbegin (
            const _Container & __cont) -> decltype(std::rbegin(__cont))   [inline], [constexpr],
[noexcept]
```

Return a reverse iterator pointing to the last element of the const container.

Parameters

__cont	Container.

References rbegin().

crend()

```
template<typename _Container>
auto std::crend (
            const _Container & __cont) -> decltype(std::rend(__cont))   [inline], [constexpr],
[noexcept]
```

Return a reverse iterator pointing one past the first element of the const container.

Parameters

__cont	Container.

References rend().

data() [1/4]

```
template<typename _Container>
auto std::data (
            _Container & __cont) -> decltype(__cont.data())   [nodiscard], [constexpr], [noexcept]
```

Return the data pointer of a container.

Parameters

__cont	Container.

Referenced by std::basic_string< _CharT, _Traits, _Alloc >::find().

data() [2/4]

```
template<typename _Tp, size_t _Nm>
_Tp * std::data (
            _Tp(&) __array[_Nm])  [nodiscard], [constexpr], [noexcept]
```

Return the data pointer of an array.

Parameters

__array	Array.

data() [3/4]

```
template<typename _Container>
auto std::data (
            const _Container & __cont) -> decltype(__cont.data())   [nodiscard], [constexpr],
[noexcept]
```

Return the data pointer of a const container.

Parameters

__cont	Container.

data() [4/4]

```
template<typename _Tp>
const _Tp * std::data (
            initializer_list< _Tp > __il)  [nodiscard], [constexpr], [noexcept]
```

Return the data pointer of an initializer list.

Parameters

_↩ _il	Initializer list.

dec()

```
ios_base & std::dec (
            ios_base & __base)  [inline]
```

Calls base.setf(ios_base::dec, ios_base::basefield).
References std::ios_base::basefield, and std::ios_base::dec.

defaultfloat()

```
ios_base & std::defaultfloat (
            ios_base & __base)  [inline]
```

Calls `base.unsetf(ios_base::floatfield)`
References std::ios_base::floatfield.

distance()

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

A generalization of pointer arithmetic.

Parameters

__first	An input iterator.
__last	An input iterator.

Returns

The distance between them.

Returns `n` such that __first + n == __last. This requires that `__last` must be reachable from `__first`. Note that `n` may be negative.
For random access iterators, this uses their + and − operations and are constant time. For other iterator classes they are linear time.
References __iterator_category(), and move().

Referenced by __merge_without_buffer(), __sample(), std::deque< _Tp, _Alloc >::_M_range_initialize(), is_heap(), is_heap_until(), is_heap_until(), std::sub_match< const char * >::length(), __gnu_parallel::multiseq_partition(), __gnu_parallel::multiseq_selection(), __gnu_pbds::detail::pat_trie_base::_Node_citer< node, leaf, head, inode, const_iterator, iterator, _A partition_point(), std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const cl std::vector< _Tp, polymorphic_allocator< _Tp > >< _BigBlock >::reserve(), std::list< _Tp, polymorphic_allocator< _Tp > >::size(), and std::list< _Tp, polymorphic_allocator< _Tp > >::splice().

dynamic_pointer_cast()

```
template<typename _Tp, typename _Tp1, _Lock_policy _Lp>
__shared_ptr< _Tp, _Lp > std::dynamic_pointer_cast (
            const __shared_ptr< _Tp1, _Lp > & __r)  [inline], [noexcept]
```

dynamic_pointer_cast

empty() [1/3]

```
template<typename _Container>
auto std::empty (
            const _Container & __cont) -> decltype(__cont.empty())   [nodiscard], [constexpr],
[noexcept]
```

Return whether a container is empty.

Parameters

__cont	Container.

Referenced by std::list< _Tp, _Alloc >::sort(), and std::list< _Tp, _Alloc >::sort().

empty() [2/3]

```
template<typename _Tp, size_t _Nm>
bool std::empty (
            const  _Tp(&)[_Nm])  [nodiscard], [constexpr], [noexcept]
```

Return whether an array is empty (always false).

empty() [3/3]

```
template<typename _Tp>
bool std::empty (
            initializer_list< _Tp > __il)  [nodiscard], [constexpr], [noexcept]
```

Return whether an initializer_list is empty.

Parameters

_↩ _il	Initializer list.

end() [1/4]

```
template<typename _Container>
auto std::end (
            _Container & __cont) -> decltype(__cont.end())   [inline], [constexpr], [noexcept]
```

Return an iterator pointing to one past the last element of the container.

Parameters

__cont	Container.

end() [2/4]

```
template<typename _Tp, size_t _Nm>
_Tp * std::end (
            _Tp(&) __arr[_Nm])  [inline], [constexpr], [noexcept]
```

Return an iterator pointing to one past the last element of the array.

Parameters

__arr	Array.

end() [3/4]

```
template<typename _Container>
auto std::end (
            const _Container & __cont) -> decltype(__cont.end())  [inline], [constexpr], [noexcept]
```

Return an iterator pointing to one past the last element of the const container.

Parameters

__cont	Container.

endl()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::endl (
            basic_ostream< _CharT, _Traits > & __os)  [inline]
```

Write a newline and flush the stream.

This manipulator is often mistakenly used when a simple newline is desired, leading to poor buffering performance. See https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html#io.↩streambuf.buffering for more on this subject.

ends()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::ends (
            basic_ostream< _CharT, _Traits > & __os)  [inline]
```

Write a null character into the output sequence.

Null character is `CharT()` by definition. For CharT of `char`, this correctly writes the ASCII `NUL` character string terminator.

fabs()

```
template<typename _Tp>
_Tp std::fabs (
            const std::complex< _Tp > & __z)  [inline]
```

fabs(__z) TR1 8.1.8 [tr.c99.cmplx.fabs]

fixed()

```
ios_base & std::fixed (
            ios_base & __base)  [inline]
```

Calls base.setf(ios_base::fixed, ios_base::floatfield).
References fixed(), std::ios_base::fixed, and std::ios_base::floatfield.
Referenced by fixed().

flush()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::flush (
            basic_ostream< _CharT, _Traits > & __os)  [inline]
```

Flushes the output stream.
This manipulator simply calls the stream's `flush()` member function.

frexp()

```
template<typename _Tp, typename _Abi, typename = __detail::__odr_helper>
enable_if_t< is_floating_point_v< _Tp >, simd< _Tp, _Abi > > frexp (
            const simd< _Tp, _Abi > & __x,
            _Samesize< int, simd< _Tp, _Abi > > * __exp)
```

splits `__v` into exponent and mantissa, the sign is kept with the mantissa
The return value will be in the range [0.5, 1.0[The `__e` value will be an integer defining the power-of-two exponent
References abs().

from_chars()

```
template<typename _Tp, enable_if_t< __or_< __is_standard_integer< _Tp >, is_same< char, remove↩
_cv_t< _Tp > > >::value, int > = 0>
from_chars_result std::from_chars (
            const char * __first,
            const char * __last,
            _Tp & __value,
            int __base = 10)  [constexpr]
```

std::from_chars for integral types.

get() [1/12]

```
template<typename _Tp, typename _Up>
const _Tp && std::get (
            const pair< _Tp, _Up > && __p)  [constexpr], [noexcept]
```

std::get overloads for accessing members of std::pair
References forward().

get() [2/12]

```
template<typename _Tp, typename _Up>
const _Tp & std::get (
            const pair< _Tp, _Up > & __p)  [constexpr], [noexcept]
```

std::get overloads for accessing members of std::pair

get() [3/12]

```
template<size_t _Int, class _Tp1, class _Tp2>
```

```
const tuple_element< _Int, pair< _Tp1, _Tp2 > >::type && std::get (
            const pair< _Tp1, _Tp2 > && __in)  [constexpr], [noexcept]
```

std::get overloads for accessing members of std::pair
References move().

get() [4/12]

```
template<size_t _Int, class _Tp1, class _Tp2>
const tuple_element< _Int, pair< _Tp1, _Tp2 > >::type & std::get (
            const pair< _Tp1, _Tp2 > & __in)  [constexpr], [noexcept]
```

std::get overloads for accessing members of std::pair

get() [5/12]

```
template<typename _Tp, typename _Up>
const _Tp && std::get (
            const pair< _Up, _Tp > && __p)  [constexpr], [noexcept]
```

std::get overloads for accessing members of std::pair
References forward().

get() [6/12]

```
template<typename _Tp, typename _Up>
const _Tp & std::get (
            const pair< _Up, _Tp > & __p)  [constexpr], [noexcept]
```

std::get overloads for accessing members of std::pair

get() [7/12]

```
template<typename _Tp, typename _Up>
_Tp && std::get (
            pair< _Tp, _Up > && __p)  [constexpr], [noexcept]
```

std::get overloads for accessing members of std::pair
References forward().

get() [8/12]

```
template<typename _Tp, typename _Up>
_Tp & std::get (
            pair< _Tp, _Up > & __p)  [constexpr], [noexcept]
```

std::get overloads for accessing members of std::pair

get() [9/12]

```
template<size_t _Int, class _Tp1, class _Tp2>
tuple_element< _Int, pair< _Tp1, _Tp2 > >::type && std::get (
            pair< _Tp1, _Tp2 > && __in)  [constexpr], [noexcept]
```

std::get overloads for accessing members of std::pair
References move().

get() [10/12]

```
template<size_t _Int, class _Tp1, class _Tp2>
tuple_element< _Int, pair< _Tp1, _Tp2 > >::type & std::get (
            pair< _Tp1, _Tp2 > & __in)  [constexpr], [noexcept]
```

std::get overloads for accessing members of std::pair

get() [11/12]

```
template<typename _Tp, typename _Up>
_Tp && std::get (
            pair< _Up, _Tp > && __p)  [constexpr], [noexcept]
```

std::get overloads for accessing members of std::pair
References forward().

get() [12/12]

```
template<typename _Tp, typename _Up>
_Tp & std::get (
            pair< _Up, _Tp > & __p)  [constexpr], [noexcept]
```

std::get overloads for accessing members of std::pair

get_money()

```
template<typename _MoneyT>
_Get_money< _MoneyT > std::get_money (
            _MoneyT & __mon,
            bool __intl = false)  [inline]
```

Extended manipulator for extracting money.

Parameters

__mon	Either long double or a specialization of `basic_string`.
__intl	A bool indicating whether international format is to be used.

Sent to a stream object, this manipulator extracts *__mon*.

get_new_handler()

```
new_handler std::get_new_handler ()  [noexcept]
```

Return the current new handler.

get_temporary_buffer()

```
template<typename _Tp>
pair< _Tp *, ptrdiff_t > std::get_temporary_buffer (
            ptrdiff_t __len)  [noexcept]
```

Allocates a temporary buffer.

Parameters

__len	The number of objects of type Tp.

Returns

See full description.

Reinventing the wheel, but this time with prettier spokes!
This function tries to obtain storage for `__len` adjacent Tp objects. The objects themselves are not constructed, of course. A pair<> is returned containing *the buffer's address and capacity (in the units of sizeof(_Tp)), or a pair of 0 values if no storage can be obtained.* Note that the capacity obtained may be less than that requested if the memory is unavailable; you should compare len with the .second return value.
Provides the nothrow exception guarantee.

get_time()

```
template<typename _CharT>
_Get_time< _CharT > std::get_time (
            std::tm * __tmb,
            const _CharT * __fmt)  [inline]
```

Extended manipulator for extracting time.
This manipulator uses time_get::get to extract time. [ext.manip]

Parameters

__tmb	struct to extract the time data to.
__fmt	format string.

getline() [1/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_istream< _CharT, _Traits > & std::getline (
            basic_istream< _CharT, _Traits > && __is,
            basic_string< _CharT, _Traits, _Alloc > & __str)  [inline]
```

Read a line from an rvalue stream into a string.
References getline().

getline() [2/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_istream< _CharT, _Traits > & std::getline (
            basic_istream< _CharT, _Traits > && __is,
            basic_string< _CharT, _Traits, _Alloc > & __str,
            _CharT __delim)  [inline]
```

Read a line from an rvalue stream into a string.
References getline().

getline() [3/6]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
basic_istream< _CharT, _Traits > & std::getline (
            basic_istream< _CharT, _Traits > & __is,
            __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > & __str)  [inline]
```

Read a line from stream into a string.

Parameters

__is	Input stream.
__str	Buffer to store into.

Returns

Reference to the input stream.

Stores characters from is into *__str* until '
' is found, the end of the stream is encountered, or str.max_size() is reached. If is.width() is non-zero, that is the limit on the number of characters stored into *__str*. Any previous contents of *__str* are erased. If end of line was encountered, it is extracted but not stored into *__str*.
References getline(), and std::basic_ios< _CharT, _Traits >::widen().

getline() [4/6]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
basic_istream< _CharT, _Traits > & std::getline (
             basic_istream< _CharT, _Traits > & __is,
             __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
             _CharT __delim)
```

Read a line from stream into a string.

Parameters

__is	Input stream.
__str	Buffer to store into.
__delim	Character marking end of line.

Returns

Reference to the input stream.

Stores characters from *__is* into *__str* until *__delim* is found, the end of the stream is encountered, or str.max_size() is reached. If is.width() is non-zero, that is the limit on the number of characters stored into *__str*. Any previous contents of *__str* are erased. If *delim* was encountered, it is extracted but not stored into *__str*.
References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append(), __gnu_cxx::__versa_string< _CharT, _Traits, _Allo
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::max_size(), std::basic_ios< _CharT, _Traits >::rdbuf(),
and std::basic_ios< _CharT, _Traits >::setstate().

getline() [5/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_istream< _CharT, _Traits > & std::getline (
             basic_istream< _CharT, _Traits > & __is,
             basic_string< _CharT, _Traits, _Alloc > & __str)  [inline]
```

Read a line from stream into a string.

Parameters

__is	Input stream.
__str	Buffer to store into.

Returns

Reference to the input stream.

Stores characters from is into *__str* until '
' is found, the end of the stream is encountered, or str.max_size() is reached. Any previous contents of *__str* are erased. If end of line is encountered, it is extracted but not stored into *__str*.
References getline(), and std::basic_ios< _CharT, _Traits >::widen().

getline() [6/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_istream< _CharT, _Traits > & std::getline (
            basic_istream< _CharT, _Traits > & __is,
            basic_string< _CharT, _Traits, _Alloc > & __str,
            _CharT __delim)
```

Read a line from stream into a string.

Parameters

__is	Input stream.
__str	Buffer to store into.
__delim	Character marking end of line.

Returns

Reference to the input stream.

Stores characters from *__is* into *__str* until *__delim* is found, the end of the stream is encountered, or str.max_size() is reached. Any previous contents of *__str* are erased. If *__delim* is encountered, it is extracted but not stored into *__str*.
References std::basic_string< _CharT, _Traits, _Alloc >::erase(), std::basic_string< _CharT, _Traits, _Alloc >::max_size(), std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().
Referenced by getline(), getline(), getline(), and getline().

hex()

```
ios_base & std::hex (
            ios_base & __base)  [inline]
```

Calls base.setf(ios_base::hex, ios_base::basefield).
References std::ios_base::basefield, hex(), and std::ios_base::hex.
Referenced by hex(), and std::regex_traits< _Ch_type >::value().

hexfloat()

```
ios_base & std::hexfloat (
            ios_base & __base)  [inline]
```

Calls base.setf(ios_base::fixed|ios_basescientific, ios_base::floatfield)
References std::ios_base::fixed, std::ios_base::floatfield, and std::ios_base::scientific.

internal()

```
ios_base & std::internal (
            ios_base & __base)  [inline]
```

Calls base.setf(ios_base::internal, ios_base::adjustfield).
References std::ios_base::adjustfield, and std::ios_base::internal.

isalnum()

```
template<typename _CharT>
bool std::isalnum (
            _CharT __c,
            const locale & __loc)  [inline]
```

Convenience interface to ctype.is(ctype_base::alnum, __c).
References use_facet().

isalpha()

```
template<typename _CharT>
bool std::isalpha (
            _CharT __c,
            const locale & __loc)  [inline]
```

Convenience interface to ctype.is(ctype_base::alpha, __c).
References use_facet().

isblank()

```
template<typename _CharT>
bool std::isblank (
            _CharT __c,
            const locale & __loc)  [inline]
```

Convenience interface to ctype.is(ctype_base::blank, __c).
References use_facet().

iscntrl()

```
template<typename _CharT>
bool std::iscntrl (
            _CharT __c,
            const locale & __loc)  [inline]
```

Convenience interface to ctype.is(ctype_base::cntrl, __c).
References use_facet().

isdigit()

```
template<typename _CharT>
bool std::isdigit (
            _CharT __c,
            const locale & __loc)  [inline]
```

Convenience interface to ctype.is(ctype_base::digit, __c).
References use_facet().

isgraph()

```
template<typename _CharT>
bool std::isgraph (
            _CharT __c,
            const locale & __loc)  [inline]
```

Convenience interface to ctype.is(ctype_base::graph, __c).
References use_facet().

islower()

```
template<typename _CharT>
bool std::islower (
            _CharT __c,
            const locale & __loc)  [inline]
```

Convenience interface to ctype.is(ctype_base::lower, __c).
References use_facet().

isprint()

```
template<typename _CharT>
bool std::isprint (
            _CharT __c,
            const locale & __loc)  [inline]
```

Convenience interface to ctype.is(ctype_base::print, __c).
References use_facet().

ispunct()

```
template<typename _CharT>
bool std::ispunct (
            _CharT __c,
            const locale & __loc)  [inline]
```

Convenience interface to ctype.is(ctype_base::punct, __c).
References use_facet().

isspace()

```
template<typename _CharT>
bool std::isspace (
            _CharT __c,
            const locale & __loc)  [inline]
```

Convenience interface to ctype.is(ctype_base::space, __c).
References use_facet().

isupper()

```
template<typename _CharT>
bool std::isupper (
            _CharT __c,
            const locale & __loc)  [inline]
```

Convenience interface to ctype.is(ctype_base::upper, __c).
References use_facet().

isxdigit()

```
template<typename _CharT>
bool std::isxdigit (
            _CharT __c,
            const locale & __loc)  [inline]
```

Convenience interface to ctype.is(ctype_base::xdigit, __c).
References use_facet().

left()

```
ios_base & std::left (
            ios_base & __base)  [inline]
```

Calls base.setf(ios_base::left, ios_base::adjustfield).
References std::ios_base::adjustfield, and std::ios_base::left.

noboolalpha()

```
ios_base & std::noboolalpha (
            ios_base & __base)  [inline]
```

Calls base.unsetf(ios_base::boolalpha).
References std::ios_base::boolalpha.

noshowbase()

```
ios_base & std::noshowbase (
            ios_base & __base)  [inline]
```

Calls base.unsetf(ios_base::showbase).
References std::ios_base::showbase.

noshowpoint()

```
ios_base & std::noshowpoint (
            ios_base & __base)  [inline]
```

Calls base.unsetf(ios_base::showpoint).
References std::ios_base::showpoint.

noshowpos()

```
ios_base & std::noshowpos (
            ios_base & __base)  [inline]
```

Calls base.unsetf(ios_base::showpos).
References std::ios_base::showpos.

noskipws()

```
ios_base & std::noskipws (
            ios_base & __base)  [inline]
```

Calls base.unsetf(ios_base::skipws).
References std::ios_base::skipws.

nounitbuf()

```
ios_base & std::nounitbuf (
            ios_base & __base)  [inline]
```

Calls base.unsetf(ios_base::unitbuf).
References std::ios_base::unitbuf.

nouppercase()

```
ios_base & std::nouppercase (
            ios_base & __base)  [inline]
```

Calls base.unsetf(ios_base::uppercase).
References std::ios_base::uppercase.

oct()

```
ios_base & std::oct (
            ios_base & __base)  [inline]
```

Calls base.setf(ios_base::oct, ios_base::basefield).
References std::ios_base::basefield, and std::ios_base::oct.
Referenced by std::regex_traits< _Ch_type >::value().

operator"!=() [1/5]

```
template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
bool std::operator!= (
            const multimap< _Key, _Tp, _Compare, _Alloc > & __x,
            const multimap< _Key, _Tp, _Compare, _Alloc > & __y)  [inline]
```

Based on operator==.

operator"!=() [2/5]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator!= (
            const multiset< _Key, _Compare, _Alloc > & __x,
            const multiset< _Key, _Compare, _Alloc > & __y)  [inline]
```

Returns !(x == y).

operator"!=() [3/5]

```
template<typename _Tp, typename _Seq>
bool std::operator!= (
            const queue< _Tp, _Seq > & __x,
            const queue< _Tp, _Seq > & __y)  [inline], [nodiscard]
```

Based on operator==.

operator"!=() [4/5]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator!= (
            const set< _Key, _Compare, _Alloc > & __x,
            const set< _Key, _Compare, _Alloc > & __y)  [inline]
```

Returns !(x == y).

operator"!=() [5/5]

```
template<typename _Tp, typename _Seq>
bool std::operator!= (
            const stack< _Tp, _Seq > & __x,
            const stack< _Tp, _Seq > & __y)  [inline], [nodiscard]
```

Based on operator==.

operator&()

```
template<size_t _Nb>
bitset< _Nb > std::operator& (
            const bitset< _Nb > & __x,
            const bitset< _Nb > & __y)  [inline], [constexpr], [noexcept]
```

Global bitwise operations on bitsets.

Parameters

_↩ _x	A bitset.
_↩ _y	A bitset of the same size as __x.

Returns

A new bitset.

These should be self-explanatory.

operator+() [1/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > std::operator+ (
            _CharT __lhs,
            const basic_string< _CharT, _Traits, _Alloc > & __rhs)  [inline], [nodiscard], [constexpr]
```

Concatenate character and string.

Parameters

__lhs	First string.
__rhs	Last string.

Returns

New string with *__lhs* followed by *__rhs*.

References std::basic_string< _CharT, _Traits, _Alloc >::c_str(), std::basic_string< _CharT, _Traits, _Alloc >::get_allocator(), and std::basic_string< _CharT, _Traits, _Alloc >::size().

operator+() [2/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > std::operator+ (
            const _CharT * __lhs,
            const basic_string< _CharT, _Traits, _Alloc > & __rhs)  [inline], [nodiscard], [constexpr]
```

Concatenate C string and string.

Parameters

__lhs	First string.
__rhs	Last string.

Returns

New string with value of *__lhs* followed by *__rhs*.

References std::basic_string< _CharT, _Traits, _Alloc >::c_str(), std::basic_string< _CharT, _Traits, _Alloc >::get_allocator(), and std::basic_string< _CharT, _Traits, _Alloc >::size().

operator+() [3/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > std::operator+ (
            const basic_string< _CharT, _Traits, _Alloc > & __lhs,
            _CharT __rhs)  [inline], [nodiscard], [constexpr]
```

Concatenate string and character.

Parameters

__lhs	First string.
__rhs	Last string.

Returns

New string with *__lhs* followed by *__rhs*.

References std::basic_string< _CharT, _Traits, _Alloc >::c_str(), std::basic_string< _CharT, _Traits, _Alloc >::get_allocator(), and std::basic_string< _CharT, _Traits, _Alloc >::size().

operator+() [4/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > std::operator+ (
            const basic_string< _CharT, _Traits, _Alloc > & __lhs,
            const _CharT * __rhs)  [inline], [nodiscard], [constexpr]
```

Concatenate string and C string.

Parameters

__lhs	First string.
__rhs	Last string.

Returns

New string with *__lhs* followed by *__rhs*.

References std::basic_string< _CharT, _Traits, _Alloc >::c_str(), std::basic_string< _CharT, _Traits, _Alloc >::get_allocator(), and std::basic_string< _CharT, _Traits, _Alloc >::size().

operator+() [5/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > std::operator+ (
            const basic_string< _CharT, _Traits, _Alloc > & __lhs,
            const basic_string< _CharT, _Traits, _Alloc > & __rhs)  [inline], [nodiscard], [constexpr]
```

Concatenate two strings.

Parameters

__lhs	First string.
__rhs	Last string.

Returns

New string with value of *__lhs* followed by *__rhs*.

References std::basic_string< _CharT, _Traits, _Alloc >::c_str(), std::basic_string< _CharT, _Traits, _Alloc >::get_allocator(), and std::basic_string< _CharT, _Traits, _Alloc >::size().

operator<() [1/5]

```
template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
bool std::operator< (
            const multimap< _Key, _Tp, _Compare, _Alloc > & __x,
            const multimap< _Key, _Tp, _Compare, _Alloc > & __y)  [inline]
```

Multimap ordering relation.

Parameters

_↩ _x	A multimap.
_↩ _y	A multimap of the same type as __x.

Returns

True iff *x* is lexicographically less than *y*.

This is a total ordering relation. It is linear in the size of the multimaps. The elements must be comparable with <. See std::lexicographical_compare() for how the determination is made.

operator<() [2/5]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator< (
            const multiset< _Key, _Compare, _Alloc > & __x,
            const multiset< _Key, _Compare, _Alloc > & __y)  [inline]
```

Multiset ordering relation.

Parameters

_↩ _x	A multiset.
_↩ _y	A multiset of the same type as __x.

Returns

True iff __x is lexicographically less than __y.

This is a total ordering relation. It is linear in the size of the sets. The elements must be comparable with <. See std::lexicographical_compare() for how the determination is made.

operator<() [3/5]

```
template<typename _Tp, typename _Seq>
bool std::operator< (
            const queue< _Tp, _Seq > & __x,
            const queue< _Tp, _Seq > & __y)  [inline], [nodiscard]
```

Queue ordering relation.

Parameters

_↩ _x	A queue.
_↩ _y	A queue of the same type as *x*.

Returns

True iff __x is lexicographically less than __y.

This is an total ordering relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, the elements must be comparable with <, and std↩ ::lexicographical_compare() is usually used to make the determination.
References std::queue< _Tp, _Sequence >::c.

operator<() [4/5]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator< (
            const set< _Key, _Compare, _Alloc > & __x,
            const set< _Key, _Compare, _Alloc > & __y)  [inline]
```

Set ordering relation.

Parameters

_↩ _x	A set.
_↩ _y	A set of the same type as *x*.

Returns

True iff __x is lexicographically less than __y.

This is a total ordering relation. It is linear in the size of the sets. The elements must be comparable with <.
See std::lexicographical_compare() for how the determination is made.

operator<() [5/5]

```
template<typename _Tp, typename _Seq>
bool std::operator< (
            const stack< _Tp, _Seq > & __x,
            const stack< _Tp, _Seq > & __y)  [inline], [nodiscard]
```

Stack ordering relation.

Parameters

_↩ _x	A stack.
_↩ _y	A stack of the same type as *x*.

Returns

True iff *x* is lexicographically less than *__y*.

This is an total ordering relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, the elements must be comparable with <, and std::lexicographical_compare() is usually used to make the determination.

operator<<() [1/24]

```
template<typename _Ostream, typename _Tp>
__rvalue_stream_insertion_t< _Ostream, _Tp > std::operator<< (
            _Ostream && __os,
            const _Tp & __x)  [inline]
```

Generic inserter for rvalue stream.

Parameters

__os	An input stream.
__x	A reference to the object being inserted.

Returns

__os

This is just a forwarding function to allow insertion to rvalue streams since they won't bind to the inserter functions that take an lvalue reference.

operator<<() [2/24]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
basic_ostream< _CharT, _Traits > & std::operator<< (
            basic_ostream< _CharT, _Traits > & __os,
            const __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > & __str)  [inline]
```

Write string to a stream.

Parameters

__os	Output stream.
__str	String to write out.

Returns

Reference to the output stream.

Output characters of *__str* into os following the same rules as for writing a C string.

operator<<() [3/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_ostream< _CharT, _Traits > & std::operator<< (
            basic_ostream< _CharT, _Traits > & __os,
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline]
```

Write string to a stream.

Parameters

__os	Output stream.
__str	String to write out.

Returns

Reference to the output stream.

Output characters of *__str* into os following the same rules as for writing a C string.

operator<<() [4/24]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::operator<< (
             basic_ostream< _CharT, _Traits > & __out,
             _CharT __c)  [inline]
```

Character inserters.

Parameters

__out	An output stream.
__c	A character.

Returns

out

Behaves like one of the formatted arithmetic inserters described in std::basic_ostream. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). _↩ _out.width(0) is then called.
If __c is of type char and the character type of the stream is not char, the character is widened before insertion.

operator<<() [5/24]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::operator<< (
             basic_ostream< _CharT, _Traits > & __out,
             char __c)  [inline]
```

Character inserters.

Parameters

__out	An output stream.
__c	A character.

Returns

out

Behaves like one of the formatted arithmetic inserters described in std::basic_ostream. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). _↩ _out.width(0) is then called.
If __c is of type char and the character type of the stream is not char, the character is widened before insertion.

operator<<() [6/24]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::operator<< (
            basic_ostream< _CharT, _Traits > & __out,
            const _CharT * __s)  [inline]
```

String inserters.

Parameters

__out	An output stream.
__s	A character string.

Returns

out

Precondition

__s must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in std::basic_ostream. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at __s, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

operator<<() [7/24]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::operator<< (
            basic_ostream< _CharT, _Traits > & __out,
            const char * __s)
```

String inserters.

Parameters

__out	An output stream.
__s	A character string.

Returns

out

Precondition

__s must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in std::basic_ostream. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at __s, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

operator<<() [8/24]

```
template<typename _Traits>
basic_ostream< char, _Traits > & std::operator<< (
            basic_ostream< char, _Traits > & ,
            char16_t )  [delete]
```

Character inserters.

Parameters

__out	An output stream.
__c	A character.

Returns

out

Behaves like one of the formatted arithmetic inserters described in std::basic_ostream. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.
If `__c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

operator<<() [9/24]

```
template<typename _Traits>
basic_ostream< char, _Traits > & std::operator<< (
            basic_ostream< char, _Traits > & ,
            char32_t )  [delete]
```

Character inserters.

Parameters

__out	An output stream.
__c	A character.

Returns

out

Behaves like one of the formatted arithmetic inserters described in std::basic_ostream. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.
If `__c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

operator<<() [10/24]

```
template<typename _Traits>
basic_ostream< char, _Traits > & std::operator<< (
            basic_ostream< char, _Traits > & ,
            const char16_t * )  [delete]
```

String inserters.

Parameters

__out	An output stream.
__s	A character string.

Returns

out

Precondition

__s must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in std::basic_ostream. After constructing a sentry object with good status, this function inserts traits::length(__s) characters starting at __s, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). __out.width(0) is then called.

operator<<() [11/24]

```
template<typename _Traits>
basic_ostream< char, _Traits > & std::operator<< (
            basic_ostream< char, _Traits > & ,
            const char32_t * )  [delete]
```

String inserters.

Parameters

__out	An output stream.
__s	A character string.

Returns

out

Precondition

__s must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in std::basic_ostream. After constructing a sentry object with good status, this function inserts traits::length(__s) characters starting at __s, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). __out.width(0) is then called.

operator<<() [12/24]

```
template<typename _Traits>
basic_ostream< char, _Traits > & std::operator<< (
            basic_ostream< char, _Traits > & ,
            const wchar_t * )  [delete]
```

String inserters.

Parameters

__out	An output stream.
__s	A character string.

Returns

out

Precondition

__s must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in std::basic_ostream. After constructing a sentry object with good status, this function inserts traits::length(__s) characters starting at __s, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). __out.width(0) is then called.

operator<<() [13/24]

```
template<typename _Traits>
basic_ostream< char, _Traits > & std::operator<< (
            basic_ostream< char, _Traits > & ,
            wchar_t )  [delete]
```

Character inserters.

Parameters

__out	An output stream.
__c	A character.

Returns

out

Behaves like one of the formatted arithmetic inserters described in std::basic_ostream. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). _↩ _out.width(0) is then called.
If __c is of type char and the character type of the stream is not char, the character is widened before insertion.

operator<<() [14/24]

```
template<typename _Traits>
basic_ostream< char, _Traits > & std::operator<< (
            basic_ostream< char, _Traits > & __out,
            char __c)  [inline]
```

Character inserters.

Parameters

__out	An output stream.
__c	A character.

Returns

out

Behaves like one of the formatted arithmetic inserters described in std::basic_ostream. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). _↩ _out.width(0) is then called.
If __c is of type char and the character type of the stream is not char, the character is widened before insertion.

operator<<() [15/24]

```
template<typename _Traits>
basic_ostream< char, _Traits > & std::operator<< (
            basic_ostream< char, _Traits > & __out,
            const char * __s)  [inline]
```

String inserters.

Parameters

__out	An output stream.
__s	A character string.

Returns

out

Precondition

__s must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in std::basic_ostream. After constructing a sentry object with good status, this function inserts traits::length(__s) characters starting at __s, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). __out.width(0) is then called.

operator<<() [16/24]

```
template<typename _Traits>
basic_ostream< char, _Traits > & std::operator<< (
            basic_ostream< char, _Traits > & __out,
            const signed char * __s)  [inline]
```

String inserters.

Parameters

__out	An output stream.
__s	A character string.

Returns

out

Precondition

__s must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in std::basic_ostream. After constructing a sentry object with good status, this function inserts traits::length(__s) characters starting at __s, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). __out.width(0) is then called.

operator<<() [17/24]

```
template<typename _Traits>
basic_ostream< char, _Traits > & std::operator<< (
            basic_ostream< char, _Traits > & __out,
            const unsigned char * __s)  [inline]
```

String inserters.

Parameters

__out	An output stream.
__s	A character string.

Returns

out

Precondition

__s must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in std::basic_ostream. After constructing a sentry object with good status, this function inserts traits::length(__s) characters starting at __s, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). __out.width(0) is then called.

operator<<() [18/24]

```
template<typename _Traits>
basic_ostream< char, _Traits > & std::operator<< (
            basic_ostream< char, _Traits > & __out,
            signed char __c)  [inline]
```

Character inserters.

Parameters

__out	An output stream.
__c	A character.

Returns

out

Behaves like one of the formatted arithmetic inserters described in std::basic_ostream. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). _↩ _out.width(0) is then called.
If __c is of type char and the character type of the stream is not char, the character is widened before insertion.

operator<<() [19/24]

```
template<typename _Traits>
basic_ostream< char, _Traits > & std::operator<< (
            basic_ostream< char, _Traits > & __out,
            unsigned char __c)  [inline]
```

Character inserters.

Parameters

__out	An output stream.
__c	A character.

Returns

out

Behaves like one of the formatted arithmetic inserters described in std::basic_ostream. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). _↩ _out.width(0) is then called.
If __c is of type char and the character type of the stream is not char, the character is widened before insertion.

operator<<() [20/24]

```
template<typename _Traits>
basic_ostream< wchar_t, _Traits > & std::operator<< (
            basic_ostream< wchar_t, _Traits > & ,
            char16_t )  [delete]
```

Character inserters.

Parameters

__out	An output stream.
__c	A character.

Returns

out

Behaves like one of the formatted arithmetic inserters described in std::basic_ostream. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.
If `__c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

operator<<() [21/24]

```
template<typename _Traits>
basic_ostream< wchar_t, _Traits > & std::operator<< (
            basic_ostream< wchar_t, _Traits > & ,
            char32_t )  [delete]
```

Character inserters.

Parameters

__out	An output stream.
__c	A character.

Returns

out

Behaves like one of the formatted arithmetic inserters described in std::basic_ostream. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.
If `__c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

operator<<() [22/24]

```
template<typename _Traits>
basic_ostream< wchar_t, _Traits > & std::operator<< (
            basic_ostream< wchar_t, _Traits > & ,
            const char16_t * )  [delete]
```

String inserters.

Parameters

__out	An output stream.
__s	A character string.

Returns

out

Precondition

`__s` must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in std::basic_ostream. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at `__s`, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

operator<<() [23/24]

```
template<typename _Traits>
basic_ostream< wchar_t, _Traits > & std::operator<< (
            basic_ostream< wchar_t, _Traits > & ,
            const char32_t * )  [delete]
```

String inserters.

Parameters

__out	An output stream.
__s	A character string.

Returns

out

Precondition

__s must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in std::basic_ostream. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at __s, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

operator<<() [24/24]

```
template<class _CharT, class _Traits, size_t _Nb>
std::basic_ostream< _CharT, _Traits > & std::operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const bitset< _Nb > & __x)
```

Global I/O operators for bitsets.

Direct I/O between streams and bitsets is supported. Output is straightforward. Input will skip whitespace, only accept *0* and *1* characters, and will only extract as many digits as the bitset will hold.

Referenced by std::shared_ptr< _State_base >::operator<<().

operator<=() [1/5]

```
template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
bool std::operator<= (
            const multimap< _Key, _Tp, _Compare, _Alloc > & __x,
            const multimap< _Key, _Tp, _Compare, _Alloc > & __y)  [inline]
```

Based on operator<.

operator<=() [2/5]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator<= (
            const multiset< _Key, _Compare, _Alloc > & __x,
            const multiset< _Key, _Compare, _Alloc > & __y)  [inline]
```

Returns !(y < x)

operator<=() [3/5]

```
template<typename _Tp, typename _Seq>
bool std::operator<= (
            const queue< _Tp, _Seq > & __x,
            const queue< _Tp, _Seq > & __y)  [inline], [nodiscard]
```

Based on operator<.

operator<=() [4/5]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator<= (
            const set< _Key, _Compare, _Alloc > & __x,
            const set< _Key, _Compare, _Alloc > & __y)  [inline]
```

Returns !(y < x)

operator<=() [5/5]

```
template<typename _Tp, typename _Seq>
bool std::operator<= (
            const stack< _Tp, _Seq > & __x,
            const stack< _Tp, _Seq > & __y)  [inline], [nodiscard]
```

Based on operator<.

operator<=>() [1/7]

```
template<typename _CharT, typename _Traits, typename _Alloc>
auto std::operator<=> (
            const basic_string< _CharT, _Traits, _Alloc > & __lhs,
            const _CharT * __rhs) -> decltype(__detail::__char_traits_cmp_cat<_Traits>(0))   [nodiscard],
[constexpr], [noexcept]
```

Three-way comparison of a string and a C string.

Parameters

__lhs	A string.
__rhs	A null-terminated string.

Returns

A value indicating whether __lhs is less than, equal to, greater than, or incomparable with __rhs.

operator<=>() [2/7]

```
template<typename _CharT, typename _Traits, typename _Alloc>
auto std::operator<=> (
            const basic_string< _CharT, _Traits, _Alloc > & __lhs,
            const basic_string< _CharT, _Traits, _Alloc > & __rhs) -> decltype(__detail::__↩
char_traits_cmp_cat<_Traits>(0))   [nodiscard], [constexpr], [noexcept]
```

Three-way comparison of a string and a C string.

Parameters

__lhs	A string.

__rhs	A null-terminated string.

Returns

A value indicating whether __lhs is less than, equal to, greater than, or incomparable with __rhs.

operator<=>() [3/7]

```
template<typename _Tp, typename _Alloc>
__detail::__synth3way_t< _Tp > std::operator<=> (
            const deque< _Tp, _Alloc > & __x,
            const deque< _Tp, _Alloc > & __y)  [inline], [nodiscard]
```

Deque ordering relation.

Parameters

_x	A deque.
_y	A deque of the same type as __x.

Returns

A value indicating whether __x is less than, equal to, greater than, or incomparable with __y.

See std::lexicographical_compare_three_way() for how the determination is made. This operator is used to synthesize relational operators like < and >= etc.
References std::deque< _Tp, _Alloc >::begin(), std::deque< _Tp, _Alloc >::end(), and std::deque< _Tp, _Alloc >::size().

operator<=>() [4/7]

```
template<typename _Tp, typename _Alloc>
__detail::__synth3way_t< _Tp > std::operator<=> (
            const forward_list< _Tp, _Alloc > & __x,
            const forward_list< _Tp, _Alloc > & __y)  [inline], [nodiscard]
```

Forward list ordering relation.

Parameters

_x	A forward_list.
_y	A forward_list of the same type as __x.

Returns

A value indicating whether __x is less than, equal to, greater than, or incomparable with __y.

See std::lexicographical_compare_three_way() for how the determination is made. This operator is used to synthesize relational operators like < and >= etc.

operator<=>() [5/7]

```
template<typename _Tp, typename _Alloc>
__detail::__synth3way_t< _Tp > std::operator<=> (
            const list< _Tp, _Alloc > & __x,
            const list< _Tp, _Alloc > & __y)  [inline], [nodiscard]
```

List ordering relation.

Parameters

_↩ _x	A list.
_↩ _y	A list of the same type as __x.

Returns

A value indicating whether __x is less than, equal to, greater than, or incomparable with __y.

See std::lexicographical_compare_three_way() for how the determination is made. This operator is used to synthesize relational operators like < and >= etc.
References std::list< _Tp, _Alloc >::begin(), std::list< _Tp, _Alloc >::end(), and std::list< _Tp, _Alloc >::size().

operator<=>() [6/7]

```
template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
__detail::__synth3way_t< pair< const _Key, _Tp > > std::operator<=> (
            const map< _Key, _Tp, _Compare, _Alloc > & __x,
            const map< _Key, _Tp, _Compare, _Alloc > & __y)  [inline]
```

Map ordering relation.

Parameters

_↩ _x	A map.
_↩ _y	A map of the same type as x.

Returns

A value indicating whether __x is less than, equal to, greater than, or incomparable with __y.

This is a total ordering relation. It is linear in the size of the maps. The elements must be comparable with <.
See std::lexicographical_compare_three_way() for how the determination is made. This operator is used to synthesize relational operators like < and >= etc.

operator<=>() [7/7]

```
template<typename _Tp, typename _Alloc>
__detail::__synth3way_t< _Tp > std::operator<=> (
            const vector< _Tp, _Alloc > & __x,
            const vector< _Tp, _Alloc > & __y)  [nodiscard], [constexpr]
```

Vector ordering relation.

Parameters

_↩ _x	A vector.
_↩ _y	A vector of the same type as __x.

Returns

A value indicating whether __x is less than, equal to, greater than, or incomparable with __y.

See std::lexicographical_compare_three_way() for how the determination is made. This operator is used to synthesize relational operators like < and >= etc.
References std::vector< _Tp, _Alloc >::begin(), std::vector< _Tp, _Alloc >::end(), and lexicographical_compare_three_way().

operator==() [1/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
bool std::operator== (
            const basic_string< _CharT, _Traits, _Alloc > & __lhs,
            const _CharT * __rhs)  [inline], [nodiscard], [constexpr]
```

Test equivalence of string and C string.

Parameters

__lhs	String.
__rhs	C string.

Returns

True if *__lhs.compare*(*__rhs*) == 0. False otherwise.

References std::basic_string< _CharT, _Traits, _Alloc >::data(), and std::basic_string< _CharT, _Traits, _Alloc >::size().

operator==() [2/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
bool std::operator== (
            const basic_string< _CharT, _Traits, _Alloc > & __lhs,
            const basic_string< _CharT, _Traits, _Alloc > & __rhs)  [inline], [nodiscard], [constexpr],
[noexcept]
```

Test equivalence of two strings.

Parameters

__lhs	First string.
__rhs	Second string.

Returns

True if *__lhs.compare*(*__rhs*) == 0. False otherwise.

operator==() [3/14]

```
template<typename _Tp, typename _Alloc>
bool std::operator== (
            const deque< _Tp, _Alloc > & __x,
            const deque< _Tp, _Alloc > & __y)  [inline], [nodiscard]
```

Deque equality comparison.

Parameters

_↩ _x	A deque.
_↩ _y	A deque of the same type as __x.

Returns

True iff the size and elements of the deques are equal.

This is an equivalence relation. It is linear in the size of the deques. Deques are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

operator==() [4/14]

```
template<typename _Tp, typename _Alloc>
bool std::operator== (
            const forward_list< _Tp, _Alloc > & __lx,
            const forward_list< _Tp, _Alloc > & __ly)
```

Forward list equality comparison.

Parameters

_↩ _lx	A forward_list
_↩ _ly	A forward_list of the same type as __lx.

Returns

True iff the elements of the forward lists are equal.

This is an equivalence relation. It is linear in the number of elements of the forward lists. Deques are considered equivalent if corresponding elements compare equal.
References std::forward_list< _Tp, _Alloc >::cbegin(), and std::forward_list< _Tp, _Alloc >::cend().

operator==() [5/14]

```
template<typename _StateT>
bool std::operator== (
            const fpos< _StateT > & __lhs,
            const fpos< _StateT > & __rhs)  [inline]
```

Test if equivalent to another position.

operator==() [6/14]

```
template<typename _Res, typename...  _Args>
bool std::operator== (
            const function< _Res(_Args...)> & __f,
            nullptr_t )  [inline], [noexcept]
```

Test whether a polymorphic function object wrapper is empty.

Returns

`true` if the wrapper has no target, `false` otherwise

This function will not throw exceptions.

operator==() [7/14]

```
template<typename _Tp, typename _Alloc>
bool std::operator== (
            const list< _Tp, _Alloc > & __x,
            const list< _Tp, _Alloc > & __y)  [inline], [nodiscard]
```

List equality comparison.

Parameters

_↩ _x	A list.
_↩ _y	A list of the same type as __x.

Returns

True iff the size and elements of the lists are equal.

This is an equivalence relation. It is linear in the size of the lists. Lists are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

operator==() [8/14]

```
template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
bool std::operator== (
            const map< _Key, _Tp, _Compare, _Alloc > & __x,
            const map< _Key, _Tp, _Compare, _Alloc > & __y)  [inline]
```

Map equality comparison.

Parameters

_↩ _x	A map.
_↩ _y	A map of the same type as *x*.

Returns

True iff the size and elements of the maps are equal.

This is an equivalence relation. It is linear in the size of the maps. Maps are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

operator==() [9/14]

```
template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
bool std::operator== (
            const multimap< _Key, _Tp, _Compare, _Alloc > & __x,
            const multimap< _Key, _Tp, _Compare, _Alloc > & __y)  [inline]
```

Multimap equality comparison.

Parameters

_↩ _x	A multimap.
_↩ _y	A multimap of the same type as __x.

Returns

True iff the size and elements of the maps are equal.

This is an equivalence relation. It is linear in the size of the multimaps. Multimaps are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

operator==() [10/14]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator== (
            const multiset< _Key, _Compare, _Alloc > & __x,
            const multiset< _Key, _Compare, _Alloc > & __y)  [inline]
```

Multiset equality comparison.

Parameters

_↩ _x	A multiset.
_↩ _y	A multiset of the same type as __x.

Returns

True iff the size and elements of the multisets are equal.

This is an equivalence relation. It is linear in the size of the multisets. Multisets are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

operator==() [11/14]

```
template<typename _Tp, typename _Seq>
bool std::operator== (
            const queue< _Tp, _Seq > & __x,
            const queue< _Tp, _Seq > & __y)  [inline], [nodiscard]
```

Queue equality comparison.

Parameters

_↩ _x	A queue.
_↩ _y	A queue of the same type as __x.

Returns

True iff the size and elements of the queues are equal.

This is an equivalence relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, and queues are considered equivalent if their sequences compare equal.
References std::queue< _Tp, _Sequence >::c.

operator==() [12/14]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator== (
            const set< _Key, _Compare, _Alloc > & __x,
            const set< _Key, _Compare, _Alloc > & __y)  [inline]
```

Set equality comparison.

Parameters

_↩ _x	A set.
_↩ _y	A set of the same type as *x*.

Returns

True iff the size and elements of the sets are equal.

This is an equivalence relation. It is linear in the size of the sets. Sets are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

operator==() [13/14]

```
template<typename _Tp, typename _Seq>
bool std::operator== (
            const stack< _Tp, _Seq > & __x,
            const stack< _Tp, _Seq > & __y)  [inline], [nodiscard]
```

Stack equality comparison.

Parameters

_↩ _x	A stack.
_↩ _y	A stack of the same type as __x.

Returns

True iff the size and elements of the stacks are equal.

This is an equivalence relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, and stacks are considered equivalent if their sequences compare equal.

operator==() [14/14]

```
template<typename _Tp, typename _Alloc>
bool std::operator== (
            const vector< _Tp, _Alloc > & __x,
            const vector< _Tp, _Alloc > & __y)  [inline], [nodiscard], [constexpr]
```

Vector equality comparison.

Parameters

_↩ _x	A vector.
_↩ _y	A vector of the same type as __x.

Returns

True iff the size and elements of the vectors are equal.

This is an equivalence relation. It is linear in the size of the vectors. Vectors are considered equivalent if their sizes are equal, and if corresponding elements compare equal.
References std::vector< _Tp, _Alloc >::begin(), std::vector< _Tp, _Alloc >::end(), and std::vector< _Tp, _Alloc >::size().

operator>() [1/5]

```
template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
bool std::operator> (
            const multimap< _Key, _Tp, _Compare, _Alloc > & __x,
            const multimap< _Key, _Tp, _Compare, _Alloc > & __y)  [inline]
```

Based on operator<.

operator>() [2/5]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator> (
            const multiset< _Key, _Compare, _Alloc > & __x,
            const multiset< _Key, _Compare, _Alloc > & __y)  [inline]
```

Returns y < x.

operator>() [3/5]

```
template<typename _Tp, typename _Seq>
bool std::operator> (
            const queue< _Tp, _Seq > & __x,
            const queue< _Tp, _Seq > & __y)  [inline], [nodiscard]
```

Based on operator<.

operator>() [4/5]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator> (
            const set< _Key, _Compare, _Alloc > & __x,
            const set< _Key, _Compare, _Alloc > & __y)  [inline]
```

Returns y < x.

operator>() [5/5]

```
template<typename _Tp, typename _Seq>
bool std::operator> (
            const stack< _Tp, _Seq > & __x,
            const stack< _Tp, _Seq > & __y)  [inline], [nodiscard]
```

Based on operator<.

operator>=() [1/5]

```
template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
bool std::operator>= (
            const multimap< _Key, _Tp, _Compare, _Alloc > & __x,
            const multimap< _Key, _Tp, _Compare, _Alloc > & __y)  [inline]
```

Based on operator<.

operator>=() [2/5]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator>= (
            const multiset< _Key, _Compare, _Alloc > & __x,
            const multiset< _Key, _Compare, _Alloc > & __y)  [inline]
```

Returns !(x < y)

operator>=() [3/5]

```
template<typename _Tp, typename _Seq>
bool std::operator>= (
            const queue< _Tp, _Seq > & __x,
            const queue< _Tp, _Seq > & __y)  [inline], [nodiscard]
```

Based on operator<.

operator>=() [4/5]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator>= (
            const set< _Key, _Compare, _Alloc > & __x,
            const set< _Key, _Compare, _Alloc > & __y)  [inline]
```

Returns !(x < y)

operator>=() [5/5]

```
template<typename _Tp, typename _Seq>
bool std::operator>= (
            const stack< _Tp, _Seq > & __x,
            const stack< _Tp, _Seq > & __y)  [inline], [nodiscard]
```

Based on operator<.

operator>>() [1/10]

```
template<typename _Istream, typename _Tp>
__rvalue_stream_extraction_t< _Istream, _Tp > std::operator>> (
            _Istream && __is,
            _Tp && __x)  [inline]
```

Generic extractor for rvalue stream.

Parameters

_↩ _is	An input stream.
_↩ _x	A reference to the extraction target.

Returns

__is

This is just a forwarding function to allow extraction from rvalue streams since they won't bind to the extractor functions that take an lvalue reference.

operator>>() [2/10]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::operator>> (
            basic_istream< _CharT, _Traits > & __in,
            _CharT & __c)
```

Character extractors.

Parameters

_↩ _in	An input stream.
_↩ _c	A character reference.

Returns

in

Behaves like one of the formatted arithmetic extractors described in std::basic_istream. After constructing a sentry object with good status, this function extracts a character (if one is available) and stores it in __c. Otherwise, sets failbit in the input stream.
References std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rd and std::basic_ios< _CharT, _Traits >::setstate().

operator>>() [3/10]

```
template<typename _CharT, typename _Traits, size_t _Num>
basic_istream< _CharT, _Traits > & std::operator>> (
            basic_istream< _CharT, _Traits > & __in,
            _CharT(&) __s[_Num])  [inline]
```

Character string extractors.

Parameters

_↩ _*in*	An input stream.
_↩ _*s*	A character array (or a pointer to an array before C++20).

Returns

__in

Behaves like one of the formatted arithmetic extractors described in `std::basic_istream`. After constructing a sentry object with good status, this function extracts up to `n` characters and stores them into the array __`s`. `n` is defined as:

- if `width()` is greater than zero, `n` is `min(width(), n)`
- otherwise `n` is the number of elements of the array
- (before C++20 the pointer is assumed to point to an array of the largest possible size for an array of `char_type`).

Characters are extracted and stored until one of the following happens:

- `n - 1` characters are stored
- EOF is reached
- the next character is whitespace according to the current locale

`width(0)` is then called for the input stream.
If no characters are extracted, sets failbit.

operator>>() [4/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
basic_istream< _CharT, _Traits > & std::operator>> (
            basic_istream< _CharT, _Traits > & __is,
            __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > & __str)
```

Read stream into a string.

Parameters

__*is*	Input stream.
__*str*	Buffer to store into.

Returns

Reference to the input stream.

Stores characters from __*is* into __*str* until whitespace is found, the end of the stream is encountered, or str.max_size() is reached. If is.width() is non-zero, that is the limit on the number of characters stored into __*str*. Any previous contents of __*str* are erased.
References std::ios_base::getloc(), std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::setstate(), use_facet(), and std::ios_base::width().

operator>>() [5/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_istream< _CharT, _Traits > & std::operator>> (
            basic_istream< _CharT, _Traits > & __is,
            basic_string< _CharT, _Traits, _Alloc > & __str)
```

Read stream into a string.

Parameters

__is	Input stream.
__str	Buffer to store into.

Returns

Reference to the input stream.

Stores characters from *__is* into *__str* until whitespace is found, the end of the stream is encountered, or str.max_size() is reached. If is.width() is non-zero, that is the limit on the number of characters stored into *__str*. Any previous contents of *__str* are erased.
References std::basic_string< _CharT, _Traits, _Alloc >::append(), std::basic_string< _CharT, _Traits, _Alloc >::erase(), std::ios_base::getloc(), std::basic_string< _CharT, _Traits, _Alloc >::max_size(), std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::setstate(), use_facet(), and std::ios_base::width().

operator>>() [6/10]

```
template<class _Traits>
basic_istream< char, _Traits > & std::operator>> (
            basic_istream< char, _Traits > & __in,
            signed char & __c)  [inline]
```

Character extractors.

Parameters

__in	An input stream.
__c	A character reference.

Returns

in

Behaves like one of the formatted arithmetic extractors described in std::basic_istream. After constructing a sentry object with good status, this function extracts a character (if one is available) and stores it in *__c*. Otherwise, sets failbit in the input stream.

operator>>() [7/10]

```
template<class _Traits, size_t _Num>
basic_istream< char, _Traits > & std::operator>> (
            basic_istream< char, _Traits > & __in,
            signed char(&) __s[_Num])  [inline]
```

Character string extractors.

Parameters

_↩ _*in*	An input stream.
_↩ _*s*	A character array (or a pointer to an array before C++20).

Returns

__in

Behaves like one of the formatted arithmetic extractors described in `std::basic_istream`. After constructing a sentry object with good status, this function extracts up to `n` characters and stores them into the array __`s`. `n` is defined as:

- if `width()` is greater than zero, `n` is `min(width(), n)`
- otherwise `n` is the number of elements of the array
- (before C++20 the pointer is assumed to point to an array of the largest possible size for an array of `char_type`).

Characters are extracted and stored until one of the following happens:

- `n - 1` characters are stored
- EOF is reached
- the next character is whitespace according to the current locale

`width(0)` is then called for the input stream.
If no characters are extracted, sets failbit.

operator>>() [8/10]

```
template<class _Traits>
basic_istream< char, _Traits > & std::operator>> (
            basic_istream< char, _Traits > & __in,
            unsigned char & __c)  [inline]
```

Character extractors.

Parameters

_↩ _*in*	An input stream.
_↩ _*c*	A character reference.

Returns

in

Behaves like one of the formatted arithmetic extractors described in std::basic_istream. After constructing a sentry object with good status, this function extracts a character (if one is available) and stores it in __*c*. Otherwise, sets failbit in the input stream.

operator>>() [9/10]

```
template<class _Traits, size_t _Num>
basic_istream< char, _Traits > & std::operator>> (
            basic_istream< char, _Traits > & __in,
            unsigned char(&) __s[_Num])  [inline]
```

Character string extractors.

Parameters

_↩ _in	An input stream.
_↩ _s	A character array (or a pointer to an array before C++20).

Returns

__in

Behaves like one of the formatted arithmetic extractors described in `std::basic_istream`. After constructing a sentry object with good status, this function extracts up to `n` characters and stores them into the array __s. `n` is defined as:

- if `width()` is greater than zero, `n` is `min(width(), n)`
- otherwise `n` is the number of elements of the array
- (before C++20 the pointer is assumed to point to an array of the largest possible size for an array of `char_type`).

Characters are extracted and stored until one of the following happens:

- `n - 1` characters are stored
- EOF is reached
- the next character is whitespace according to the current locale

`width(0)` is then called for the input stream.
If no characters are extracted, sets failbit.

operator>>() [10/10]

```
template<class _CharT, class _Traits, size_t _Nb>
std::basic_istream< _CharT, _Traits > & std::operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            bitset< _Nb > & __x)
```

Global I/O operators for bitsets.
Direct I/O between streams and bitsets is supported. Output is straightforward. Input will skip whitespace, only accept *0* and *1* characters, and will only extract as many digits as the bitset will hold.

operator^()

```
template<size_t _Nb>
bitset< _Nb > std::operator^ (
            const bitset< _Nb > & __x,
            const bitset< _Nb > & __y)  [inline], [constexpr], [noexcept]
```

Global bitwise operations on bitsets.

Parameters

_↩ _x	A bitset.
_↩ _y	A bitset of the same size as __x.

Returns

A new bitset.

These should be self-explanatory.

operator"" | ()

```
template<size_t _Nb>
bitset< _Nb > std::operator| (
            const bitset< _Nb > & __x,
            const bitset< _Nb > & __y)  [inline], [constexpr], [noexcept]
```

Global bitwise operations on bitsets.

Parameters

_↩ _x	A bitset.
_↩ _y	A bitset of the same size as __x.

Returns

A new bitset.

These should be self-explanatory.

put_money()

```
template<typename _MoneyT>
_Put_money< _MoneyT > std::put_money (
            const _MoneyT & __mon,
            bool __intl = false)  [inline]
```

Extended manipulator for inserting money.

Parameters

__mon	Either long double or a specialization of `basic_string`.
__intl	A bool indicating whether international format is to be used.

Sent to a stream object, this manipulator inserts *__mon*.

put_time()

```
template<typename _CharT>
_Put_time< _CharT > std::put_time (
```

```
            const std::tm * __tmb,
            const _CharT * __fmt)  [inline]
```

Extended manipulator for formatting time.

This manipulator uses time_put::put to format time. [ext.manip]

Parameters

__tmb	struct tm time data to format.
__fmt	format string.

rbegin() [1/4]

```
template<typename _Container>
auto std::rbegin (
            _Container & __cont) -> decltype(__cont.rbegin())   [inline], [constexpr], [noexcept]
```

Return a reverse iterator pointing to the last element of the container.

Parameters

__cont	Container.

Referenced by crbegin().

rbegin() [2/4]

```
template<typename _Tp, size_t _Nm>
reverse_iterator< _Tp * > std::rbegin (
            _Tp(&) __arr[_Nm])  [inline], [constexpr], [noexcept]
```

Return a reverse iterator pointing to the last element of the array.

Parameters

__arr	Array.

rbegin() [3/4]

```
template<typename _Container>
auto std::rbegin (
            const _Container & __cont) -> decltype(__cont.rbegin())   [inline], [constexpr],
[noexcept]
```

Return a reverse iterator pointing to the last element of the const container.

Parameters

__cont	Container.

rbegin() [4/4]

```
template<typename _Tp>
reverse_iterator< const _Tp * > std::rbegin (
            initializer_list< _Tp > __il)  [inline], [constexpr], [noexcept]
```

Return a reverse iterator pointing to the last element of the initializer_list.

Parameters

_↩ _il	initializer_list.

rend() [1/4]

```
template<typename _Container>
auto std::rend (
            _Container & __cont) -> decltype(__cont.rend())   [inline], [constexpr], [noexcept]
```

Return a reverse iterator pointing one past the first element of the container.

Parameters

__cont	Container.

Referenced by crend().

rend() [2/4]

```
template<typename _Tp, size_t _Nm>
reverse_iterator< _Tp * > std::rend (
            _Tp(&) __arr[_Nm])  [inline], [constexpr], [noexcept]
```

Return a reverse iterator pointing one past the first element of the array.

Parameters

__arr	Array.

rend() [3/4]

```
template<typename _Container>
auto std::rend (
            const _Container & __cont) -> decltype(__cont.rend())   [inline], [constexpr], [noexcept]
```

Return a reverse iterator pointing one past the first element of the const container.

Parameters

__cont	Container.

rend() [4/4]

```
template<typename _Tp>
reverse_iterator< const _Tp * > std::rend (
            initializer_list< _Tp > __il)  [inline], [constexpr], [noexcept]
```

Return a reverse iterator pointing one past the first element of the initializer_list.

Parameters

_↩ _il	initializer_list.

replace_copy()

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

Copy a sequence, replacing each element of one value with another value.

Parameters

__first	An input iterator.
__last	An input iterator.
__result	An output iterator.
__old_value	The value to be replaced.
__new_value	The replacement value.

Returns

The end of the output sequence, `result+`(last-first).

Copies each element in the input range [__first,__last) to the output range [__result,__result+(__last-__first)) replacing elements equal to `__old_value` with `__new_value`.

resetiosflags()

```
_Resetiosflags std::resetiosflags (
            ios_base::fmtflags __mask)  [inline]
```

Manipulator for `setf`.

Parameters

__mask	A format flags mask.

Sent to a stream object, this manipulator resets the specified flags, via *stream.setf(0,__mask)*.

return_temporary_buffer()

```
template<typename _Tp>
void std::return_temporary_buffer (
            _Tp * __p)  [inline]
```

The companion to get_temporary_buffer().

Parameters

__p	A buffer previously allocated by get_temporary_buffer.

Returns

None.

Frees the memory pointed to by __p.

right()

```
ios_base & std::right (
            ios_base & __base)  [inline]
```

Calls base.setf(ios_base::right, ios_base::adjustfield).
References std::ios_base::adjustfield, and std::ios_base::right.

sample()

```
template<typename _PopulationIterator, typename _SampleIterator, typename _Distance, typename _↩
UniformRandomBitGenerator>
_SampleIterator std::sample (
            _PopulationIterator __first,
            _PopulationIterator __last,
            _SampleIterator __out,
            _Distance __n,
            _UniformRandomBitGenerator && __g)
```

Take a random sample from a population.
References forward().

scientific()

```
ios_base & std::scientific (
            ios_base & __base)  [inline]
```

Calls base.setf(ios_base::scientific, ios_base::floatfield).
References std::ios_base::floatfield, std::ios_base::scientific, and scientific().
Referenced by scientific().

search()

```
template<typename _ForwardIterator, typename _Searcher>
_ForwardIterator std::search (
            _ForwardIterator __first,
            _ForwardIterator __last,
            const _Searcher & __searcher)  [inline], [nodiscard], [constexpr]
```

Search a sequence using a Searcher object.

Parameters

__first	A forward iterator.
__last	A forward iterator.
__searcher	A callable object.

Returns

`__searcher(__first,__last)`.first

set_new_handler()

```
new_handler std::set_new_handler (
            new_handler ) throw ( )
```

Takes a replacement handler as the argument, returns the previous handler.

setbase()

```
_Setbase std::setbase (
            int __base)  [inline]
```

Manipulator for `setf`.

Parameters

__base	A numeric base.

Sent to a stream object, this manipulator changes the `ios_base::basefield` flags to `oct`, `dec`, or `hex` when *base* is 8, 10, or 16, accordingly, and to 0 if *__base* is any other value.

setfill()

```
template<typename _CharT>
_Setfill< _CharT > std::setfill (
            _CharT __c)  [inline]
```

Manipulator for `fill`.

Parameters

_↩ _c	The new fill character.

Sent to a stream object, this manipulator calls `fill(__c)` for that object.

setiosflags()

```
_Setiosflags std::setiosflags (
            ios_base::fmtflags __mask)  [inline]
```

Manipulator for `setf`.

Parameters

__mask	A format flags mask.

Sent to a stream object, this manipulator sets the format flags to *__mask*.

setprecision()

```
_Setprecision std::setprecision (
            int __n)  [inline]
```

Manipulator for `precision`.

Parameters

_↩ _n	The new precision.

Sent to a stream object, this manipulator calls `precision(__n)` for that object.

setw()

```
_Setw std::setw (
            int __n)  [inline]
```

Manipulator for `width`.

Parameters

_↩ _n	The new width.

Sent to a stream object, this manipulator calls `width(__n)` for that object.

showbase()

```
ios_base & std::showbase (
            ios_base & __base)  [inline]
```

Calls base.setf(ios_base::showbase).
References std::ios_base::showbase.

showpoint()

```
ios_base & std::showpoint (
            ios_base & __base)  [inline]
```

Calls base.setf(ios_base::showpoint).
References std::ios_base::showpoint.

showpos()

```
ios_base & std::showpos (
            ios_base & __base)  [inline]
```

Calls base.setf(ios_base::showpos).
References std::ios_base::showpos.

size() [1/2]

```
template<typename _Container>
auto std::size (
            const _Container & __cont) -> decltype(__cont.size())   [nodiscard], [constexpr],
[noexcept]
```

Return the size of a container.

Parameters

__cont	Container.

Referenced by std::deque< _Tp, _Alloc >::_M_new_elements_at_back(), std::deque< _Tp, _Alloc >::_M_new_elements_at_front(), std::deque< _Tp, _Alloc >::_M_push_back_aux(), std::deque< _Tp, _Alloc >::_M_push_front_aux(), std::basic_string< _CharT, _Traits, std::basic_string< _CharT, _Traits, _Alloc >::find(), std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of(), std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of(), std::basic_string< _CharT, _Traits, _Alloc >::find_first_of(), std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of(), std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of(), std::basic_string< _CharT, _Traits, _Alloc >::find_last_of(), std::deque< _Tp, _Alloc >::operator=(), std::vector< _Tp, _Alloc >::operator std::basic_string< _CharT, _Traits, _Alloc >::reserve(), std::vector< _Tp, _Alloc >::reserve(), std::vector< _Tp, polymorphic_allocator< std::basic_string< _CharT, _Traits, _Alloc >::rfind(), and std::basic_string< _CharT, _Traits, _Alloc >::rfind().

size() [2/2]

```
template<typename _Tp, size_t _Nm>
size_t std::size (
            const  _Tp(&)[_Nm])  [nodiscard], [constexpr], [noexcept]
```

Return the size of an array.

skipws()

```
ios_base & std::skipws (
            ios_base & __base)  [inline]
```

Calls base.setf(ios_base::skipws).
References std::ios_base::skipws.

static_pointer_cast()

```
template<typename _Tp, typename _Tp1, _Lock_policy _Lp>
__shared_ptr< _Tp, _Lp > std::static_pointer_cast (
            const __shared_ptr< _Tp1, _Lp > & __r)  [inline], [noexcept]
```

static_pointer_cast

swap() [1/19]

```
template<class _CharT, class _Traits>
void std::swap (
            basic_filebuf< _CharT, _Traits > & __x,
            basic_filebuf< _CharT, _Traits > & __y)  [inline]
```

Swap specialization for filebufs.

swap() [2/19]

```
template<class _CharT, class _Traits>
void std::swap (
            basic_fstream< _CharT, _Traits > & __x,
            basic_fstream< _CharT, _Traits > & __y)  [inline]
```

Swap specialization for fstreams.

swap() [3/19]

```
template<class _CharT, class _Traits>
void std::swap (
            basic_ifstream< _CharT, _Traits > & __x,
            basic_ifstream< _CharT, _Traits > & __y)  [inline]
```

Swap specialization for ifstreams.

swap() [4/19]

```
template<class _CharT, class _Traits, class _Allocator>
void std::swap (
            basic_istringstream< _CharT, _Traits, _Allocator > & __x,
            basic_istringstream< _CharT, _Traits, _Allocator > & __y)  [inline]
```

Swap specialization for istringstreams.

swap() [5/19]

```
template<class _CharT, class _Traits>
void std::swap (
            basic_ofstream< _CharT, _Traits > & __x,
            basic_ofstream< _CharT, _Traits > & __y)  [inline]
```

Swap specialization for ofstreams.

swap() [6/19]

```
template<class _CharT, class _Traits, class _Allocator>
void std::swap (
            basic_ostringstream< _CharT, _Traits, _Allocator > & __x,
            basic_ostringstream< _CharT, _Traits, _Allocator > & __y)  [inline]
```

Swap specialization for ostringstreams.

swap() [7/19]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::swap (
            basic_string< _CharT, _Traits, _Alloc > & __lhs,
            basic_string< _CharT, _Traits, _Alloc > & __rhs)  [inline], [constexpr], [noexcept]
```

Swap contents of two strings.

Parameters

__lhs	First string.
__rhs	Second string.

Exchanges the contents of *__lhs* and *__rhs* in constant time.

swap() [8/19]

```
template<class _CharT, class _Traits, class _Allocator>
void std::swap (
            basic_stringbuf< _CharT, _Traits, _Allocator > & __x,
            basic_stringbuf< _CharT, _Traits, _Allocator > & __y)  [inline], [noexcept]
```

Swap specialization for stringbufs.

swap() [9/19]

```
template<class _CharT, class _Traits, class _Allocator>
void std::swap (
            basic_stringstream< _CharT, _Traits, _Allocator > & __x,
            basic_stringstream< _CharT, _Traits, _Allocator > & __y)  [inline]
```

Swap specialization for stringstreams.

swap() [10/19]

```
template<typename _Tp, typename _Alloc>
void std::swap (
            deque< _Tp, _Alloc > & __x,
            deque< _Tp, _Alloc > & __y)  [inline], [noexcept]
```

See std::deque::swap().

swap() [11/19]

```
template<typename _Tp, typename _Alloc>
void std::swap (
            forward_list< _Tp, _Alloc > & __lx,
            forward_list< _Tp, _Alloc > & __ly)  [inline], [noexcept]
```

See std::forward_list::swap().

swap() [12/19]

```
template<typename _Res, typename...  _Args>
void std::swap (
            function< _Res(_Args...)> & __x,
            function< _Res(_Args...)> & __y)  [inline], [noexcept]
```

Swap the targets of two polymorphic function object wrappers.
This function will not throw exceptions.

swap() [13/19]

```
template<typename _Tp, typename _Alloc>
void std::swap (
            list< _Tp, _Alloc > & __x,
            list< _Tp, _Alloc > & __y)  [inline], [noexcept]
```

See std::list::swap().

swap() [14/19]

```
template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
void std::swap (
            map< _Key, _Tp, _Compare, _Alloc > & __x,
            map< _Key, _Tp, _Compare, _Alloc > & __y)  [inline], [noexcept]
```

See std::map::swap().
References std::pair< _T1, _T2 >::swap().

swap() [15/19]

```
template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
void std::swap (
            multimap< _Key, _Tp, _Compare, _Alloc > & __x,
            multimap< _Key, _Tp, _Compare, _Alloc > & __y)  [inline], [noexcept]
```

See std::multimap::swap().
References std::multimap< _Key, _Tp, _Compare, _Alloc >::swap().

swap() [16/19]

```
template<typename _Key, typename _Compare, typename _Alloc>
void std::swap (
            multiset< _Key, _Compare, _Alloc > & __x,
            multiset< _Key, _Compare, _Alloc > & __y)  [inline], [noexcept]
```

See std::multiset::swap().
References std::multiset< _Key, _Compare, _Alloc >::swap().

swap() [17/19]

```
template<typename _Key, typename _Compare, typename _Alloc>
void std::swap (
            set< _Key, _Compare, _Alloc > & __x,
            set< _Key, _Compare, _Alloc > & __y)  [inline], [noexcept]
```

See std::set::swap().
References std::set< _Key, _Compare, _Alloc >::swap().

swap() [18/19]

```
template<typename _Tp, typename _Alloc>
void std::swap (
            vector< _Tp, _Alloc > & __x,
            vector< _Tp, _Alloc > & __y)  [inline], [constexpr], [noexcept]
```

See std::vector::swap().

tolower()

```
template<typename _CharT>
_CharT std::tolower (
            _CharT __c,
            const locale & __loc)  [inline]
```

Convenience interface to ctype.tolower(__c).
References tolower(), and use_facet().
Referenced by tolower().

toupper()

```
template<typename _CharT>
_CharT std::toupper (
            _CharT __c,
            const locale & __loc)  [inline]
```

Convenience interface to ctype.toupper(__c).
References toupper(), and use_facet().
Referenced by toupper().

unitbuf()

```
ios_base & std::unitbuf (
            ios_base & __base)  [inline]
```

Calls base.setf(ios_base::unitbuf).
References std::ios_base::unitbuf.

uppercase()

```
ios_base & std::uppercase (
            ios_base & __base)  [inline]
```

Calls base.setf(ios_base::uppercase).
References std::ios_base::uppercase.

ws()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::ws (
            basic_istream< _CharT, _Traits > & __is)
```

Quick and easy way to eat whitespace.
This manipulator extracts whitespace characters, stopping when the next character is non-whitespace, or when the input sequence is empty. If the sequence is empty, `eofbit` is set in the stream, but not `failbit`.
The current locale is used to distinguish whitespace characters.
Example:

```
MyClass  mc;

std::cin » std::ws » mc;
```

will skip leading whitespace before calling operator>> on cin and your object. Note that the same effect can be achieved by creating a std::basic_istream::sentry inside your definition of operator>>.
References std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::getloc(), std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::
std::basic_ios< _CharT, _Traits >::setstate(), and use_facet().

4.10.5 Variable Documentation

cerr

```
ostream std::cerr  [extern]
```
Linked to standard error (unbuffered)

cin

```
istream std::cin  [extern]
```
Linked to standard input.

clog

```
ostream std::clog  [extern]
```
Linked to standard error (buffered)

cout

```
ostream std::cout  [extern]
```
Linked to standard output.

default_sentinel

```
default_sentinel_t std::default_sentinel  [inline], [constexpr]
```
A default sentinel value.

destroying_delete

```
destroying_delete_t std::destroying_delete  [inline], [constexpr]
```
Tag variable of type destroying_delete_t.

ignore

```
_Swallow_assign std::ignore  [inline], [constexpr]
```
Used with `std::tie` to ignore an element of a tuple
When using `std::tie` to assign the elements of a tuple to variables, unwanted elements can be ignored by using `std::ignore`. For example:

```
int x, y;
std::tie(x, std::ignore, y) = std::make_tuple(1, 2, 3);
```

This assignment will perform `x=1; std::ignore=2; y=3;` which results in the second element being ignored.

Since

C++11

wcerr

```
wostream std::wcerr  [extern]
```
Linked to standard error (unbuffered)

wcin

wistream std::wcin [extern]
Linked to standard input.

wclog

wostream std::wclog [extern]
Linked to standard error (buffered)

wcout

wostream std::wcout [extern]
Linked to standard output.

4.11 std::__debug Namespace Reference

Classes

- class bitset
- class deque
- class forward_list
- class list
- class map
- class multimap
- class multiset
- class set
- class unordered_map
- class unordered_multimap
- class unordered_multiset
- class unordered_set
- class vector

Functions

- template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  deque (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> deque< _ValT, _Allocator >
- template<typename _Tp, typename _Allocator = allocator<_Tp>, typename = _RequireAllocator<_Allocator>>
  deque (size_t, _Tp, _Allocator=_Allocator()) -> deque< _Tp, _Allocator >
- template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  forward_list (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> forward_list< _ValT, _Allocator >
- template<typename _Tp, typename _Allocator = allocator<_Tp>, typename = _RequireAllocator<_Allocator>>
  forward_list (size_t, _Tp, _Allocator=_Allocator()) -> forward_list< _Tp, _Allocator >
- template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  list (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> list< _ValT, _Allocator >
- template<typename _Tp, typename _Allocator = allocator<_Tp>, typename = _RequireAllocator<_Allocator>>
  list (size_t, _Tp, _Allocator=_Allocator()) -> list< _Tp, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  map (_InputIterator, _InputIterator, _Allocator) -> map< __iter_key_t< _InputIterator >, __iter_val_t< _Input↩ Iterator >, less< __iter_key_t< _InputIterator > >, _Allocator >

- template<typename _InputIterator, typename _Compare = less<__iter_key_t<_InputIterator>>, typename _Allocator = allocator<__iter_to_alloc_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  map (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Compare, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  map (initializer_list< pair< _Key, _Tp > >, _Allocator) -> map< _Key, _Tp, less< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Compare = less<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  map (initializer_list< pair< _Key, _Tp > >, _Compare=_Compare(), _Allocator=_Allocator()) -> map< _Key, _Tp, _Compare, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  multimap (_InputIterator, _InputIterator, _Allocator) -> multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, less< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Compare = less<__iter_key_t<_InputIterator>>, typename _Allocator = allocator<__iter_to_alloc_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  multimap (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Compare, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  multimap (initializer_list< pair< _Key, _Tp > >, _Allocator) -> multimap< _Key, _Tp, less< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Compare = less<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  multimap (initializer_list< pair< _Key, _Tp > >, _Compare=_Compare(), _Allocator=_Allocator()) -> multimap< _Key, _Tp, _Compare, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  multiset (_InputIterator, _InputIterator, _Allocator) -> multiset< typename iterator_traits< _InputIterator >::value_type, less< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Compare = less<typename iterator_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  multiset (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> multiset< typename iterator_traits< _InputIterator >::value_type, _Compare, _Allocator >
- template<typename _Key, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  multiset (initializer_list< _Key >, _Allocator) -> multiset< _Key, less< _Key >, _Allocator >
- template<typename _Key, typename _Compare = less<_Key>, typename _Allocator = allocator<_Key>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  multiset (initializer_list< _Key >, _Compare=_Compare(), _Allocator=_Allocator()) -> multiset< _Key, _Compare, _Allocator >
- template<size_t _Nb>
  constexpr bitset< _Nb > **operator&** (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept
- template<typename _CharT, typename _Traits, size_t _Nb>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const bitset< _Nb > &__x)
- template<typename _Tp, typename _Alloc>
  constexpr __detail::__synth3way_t< _Tp > **operator**<=> (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)
- template<typename _Tp, typename _Alloc>
  constexpr __detail::__synth3way_t< _Tp > **operator**<=> (const forward_list< _Tp, _Alloc > &__x, const forward_list< _Tp, _Alloc > &__y)

- template<typename _Tp, typename _Alloc>
  constexpr __detail::__synth3way_t< _Tp > **operator**<=> (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  __detail::__synth3way_t< pair< const _Key, _Tp > > **operator**<=> (const map< _Key, _Tp, _Compare, _Alloc > &__lhs, const map< _Key, _Tp, _Compare, _Alloc > &__rhs)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  __detail::__synth3way_t< pair< const _Key, _Tp > > **operator**<=> (const multimap< _Key, _Tp, _Compare, _Alloc > &__lhs, const multimap< _Key, _Tp, _Compare, _Alloc > &__rhs)
- template<typename _Key, typename _Compare, typename _Alloc>
  __detail::__synth3way_t< _Key > **operator**<=> (const multiset< _Key, _Compare, _Alloc > &__lhs, const multiset< _Key, _Compare, _Alloc > &__rhs)
- template<typename _Key, typename _Compare, typename _Alloc>
  __detail::__synth3way_t< _Key > **operator**<=> (const set< _Key, _Compare, _Alloc > &__lhs, const set< _Key, _Compare, _Alloc > &__rhs)
- template<typename _Tp, typename _Alloc>
  constexpr __detail::__synth3way_t< _Tp > **operator**<=> (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)
- template<typename _Tp, typename _Alloc>
  bool **operator==** (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)
- template<typename _Tp, typename _Alloc>
  bool **operator==** (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)
- template<typename _Tp, typename _Alloc>
  bool **operator==** (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)
- template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>
  bool **operator==** (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>
  bool **operator==** (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key, typename _Compare, typename _Allocator>
  bool **operator==** (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _↩ Allocator > &__rhs)
- template<typename _Key, typename _Compare, typename _Allocator>
  bool **operator==** (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>
  bool **operator==** (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>
  bool **operator==** (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>
  bool **operator==** (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)
- template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>
  bool **operator==** (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)
- template<typename _Tp, typename _Alloc>
  constexpr bool **operator==** (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)
- template<typename _CharT, typename _Traits, size_t _Nb>
  std::basic_istream< _CharT, _Traits > & **operator>>** (std::basic_istream< _CharT, _Traits > &__is, bitset< _Nb > &__x)

- template<size_t _Nb>
  constexpr bitset< _Nb > **operator^** (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept
- template<size_t _Nb>
  constexpr bitset< _Nb > **operator|** (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  set (_InputIterator, _InputIterator, _Allocator) -> set< typename iterator_traits< _InputIterator >::value_type, less< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Compare = less<typename iterator_traits<_InputIterator>::value_type>, typename _↩ Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  set (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> set< typename iterator_traits< _InputIterator >::value_type, _Compare, _Allocator >
- template<typename _Key, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  set (initializer_list< _Key >, _Allocator) -> set< _Key, less< _Key >, _Allocator >
- template<typename _Key, typename _Compare = less<_Key>, typename _Allocator = allocator<_Key>, typename = _RequireNot↩ Allocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  set (initializer_list< _Key >, _Compare=_Compare(), _Allocator=_Allocator()) -> set< _Key, _Compare, _↩ Allocator >
- template<typename _Tp, typename _Alloc>
  void **swap** (deque< _Tp, _Alloc > &__lhs, deque< _Tp, _Alloc > &__rhs) noexcept(/*conditional */)
- template<typename _Tp, typename _Alloc>
  void swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly) noexcept(noexcept(__lx.↩ swap(__ly)))
- template<typename _Tp, typename _Alloc>
  void **swap** (list< _Tp, _Alloc > &__lhs, list< _Tp, _Alloc > &__rhs) noexcept(/*conditional */)
- template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>
  void **swap** (map< _Key, _Tp, _Compare, _Allocator > &__lhs, map< _Key, _Tp, _Compare, _Allocator > &__rhs) noexcept(/*conditional */)
- template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>
  void **swap** (multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, multimap< _Key, _Tp, _Compare, _Allocator > &__rhs) noexcept(/*conditional */)
- template<typename _Key, typename _Compare, typename _Allocator>
  void **swap** (multiset< _Key, _Compare, _Allocator > &__x, multiset< _Key, _Compare, _Allocator > &__↩ y) noexcept(/*conditional */)
- template<typename _Key, typename _Compare, typename _Allocator>
  void **swap** (set< _Key, _Compare, _Allocator > &__x, set< _Key, _Compare, _Allocator > &__↩ y) noexcept(/*conditional */)
- template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>
  void **swap** (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>
  void **swap** (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>
  void **swap** (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>
  void **swap** (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, unordered_set< _Value, _Hash, _Pred, _↩ Alloc > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename _Tp, typename _Alloc>
  constexpr void **swap** (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &__rhs) noexcept(/*conditional */)

- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  unordered_map (_InputIterator, _InputIterator, _Allocator) -> unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator > >, equal_to< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  unordered_map (_InputIterator, _InputIterator, typename unordered_map< int, int >::size_type, _Allocator) -> unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator > >, equal_to< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  unordered_map (_InputIterator, _InputIterator, typename unordered_map< int, int >::size_type, _Hash, ↩ _Allocator) -> unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Hash, equal_to< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Hash = hash<__iter_key_t<_InputIterator>>, typename _Pred = equal_to<__iter_key_↩ t<_InputIterator>>, typename _Allocator = allocator<__iter_to_alloc_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_↩ Allocator>>
  unordered_map (_InputIterator, _InputIterator, typename unordered_map< int, int >::size_type={}, _Hash=_↩ Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_map< __iter_key_t< _InputIterator >, __iter_↩ val_t< _InputIterator >, _Hash, _Pred, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  unordered_map (initializer_list< pair< _Key, _Tp > >, _Allocator) -> unordered_map< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  unordered_map (initializer_list< pair< _Key, _Tp > >, typename unordered_map< int, int >::size_type, _↩ Allocator) -> unordered_map< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  unordered_map (initializer_list< pair< _Key, _Tp > >, typename unordered_map< int, int >::size_type, _Hash, _Allocator) -> unordered_map< _Key, _Tp, _Hash, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  unordered_map (initializer_list< pair< _Key, _Tp > >, typename unordered_map< int, int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_map< _Key, _Tp, _Hash, _Pred, _↩ Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  unordered_multimap (_InputIterator, _InputIterator, _Allocator) -> unordered_multimap< __iter_key_t< _↩ InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator > >, equal_to< __iter_↩ key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  unordered_multimap (_InputIterator, _InputIterator, unordered_multimap< int, int >::size_type, _Allocator) -> unordered_multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator > >, equal_to< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = ↩ _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  unordered_multimap (_InputIterator, _InputIterator, unordered_multimap< int, int >::size_type, _Hash, _↩ Allocator) -> unordered_multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Hash, equal_to< __iter_key_t< _InputIterator > >, _Allocator >

- template<typename _InputIterator, typename _Hash = hash<__iter_key_t<_InputIterator>>, typename _Pred = equal_to<__iter_key_t<_InputIterator>>, typename _Allocator = allocator<__iter_to_alloc_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  unordered_multimap (_InputIterator, _InputIterator, unordered_multimap< int, int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Hash, _Pred, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  unordered_multimap (initializer_list< pair< _Key, _Tp > >, _Allocator) -> unordered_multimap< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  unordered_multimap (initializer_list< pair< _Key, _Tp > >, unordered_multimap< int, int >::size_type, _Allocator) -> unordered_multimap< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  unordered_multimap (initializer_list< pair< _Key, _Tp > >, unordered_multimap< int, int >::size_type, _Hash, _Allocator) -> unordered_multimap< _Key, _Tp, _Hash, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  unordered_multimap (initializer_list< pair< _Key, _Tp > >, unordered_multimap< int, int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_multimap< _Key, _Tp, _Hash, _Pred, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  unordered_multiset (_InputIterator, _InputIterator, _Allocator) -> unordered_multiset< typename iterator_traits< _InputIterator >::value_type, hash< typename iterator_traits< _InputIterator >::value_type >, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  unordered_multiset (_InputIterator, _InputIterator, unordered_multiset< int >::size_type, _Allocator) -> unordered_multiset< typename iterator_traits< _InputIterator >::value_type, hash< typename iterator_traits< _InputIterator >::value_type >, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  unordered_multiset (_InputIterator, _InputIterator, unordered_multiset< int >::size_type, _Hash, _Allocator) -> unordered_multiset< typename iterator_traits< _InputIterator >::value_type, _Hash, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Hash = hash<typename iterator_traits<_InputIterator>::value_type>, typename _Pred = equal_to<typename iterator_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  unordered_multiset (_InputIterator, _InputIterator, unordered_multiset< int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_multiset< typename iterator_traits< _InputIterator >::value_type, _Hash, _Pred, _Allocator >
- template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  unordered_multiset (initializer_list< _Tp >, _Allocator) -> unordered_multiset< _Tp, hash< _Tp >, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  unordered_multiset (initializer_list< _Tp >, unordered_multiset< int >::size_type, _Allocator) -> unordered_multiset< _Tp, hash< _Tp >, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>

unordered_multiset (initializer_list< _Tp >, unordered_multiset< int >::size_type, _Hash, _Allocator) -> unordered_multiset< _Tp, _Hash, equal_to< _Tp >, _Allocator >

- template<typename _Tp, typename _Hash = hash<_Tp>, typename _Pred = equal_to<_Tp>, typename _Allocator = allocator<_Tp>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  unordered_multiset (initializer_list< _Tp >, unordered_multiset< int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_multiset< _Tp, _Hash, _Pred, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  unordered_set (_InputIterator, _InputIterator, _Allocator) -> unordered_set< typename iterator_traits< _InputIterator >::value_type, hash< typename iterator_traits< _InputIterator >::value_type >, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  unordered_set (_InputIterator, _InputIterator, unordered_set< int >::size_type, _Allocator) -> unordered_set< typename iterator_traits< _InputIterator >::value_type, hash< typename iterator_traits< _InputIterator >::value_type >, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  unordered_set (_InputIterator, _InputIterator, unordered_set< int >::size_type, _Hash, _Allocator) -> unordered_set< typename iterator_traits< _InputIterator >::value_type, _Hash, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Hash = hash<typename iterator_traits<_InputIterator>::value_type>, typename _Pred = equal_to<typename iterator_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  unordered_set (_InputIterator, _InputIterator, unordered_set< int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_set< typename iterator_traits< _InputIterator >::value_type, _Hash, _Pred, _Allocator >
- template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  unordered_set (initializer_list< _Tp >, _Allocator) -> unordered_set< _Tp, hash< _Tp >, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  unordered_set (initializer_list< _Tp >, unordered_set< int >::size_type, _Allocator) -> unordered_set< _Tp, hash< _Tp >, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  unordered_set (initializer_list< _Tp >, unordered_set< int >::size_type, _Hash, _Allocator) -> unordered_set< _Tp, _Hash, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Hash = hash<_Tp>, typename _Pred = equal_to<_Tp>, typename _Allocator = allocator<_Tp>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  unordered_set (initializer_list< _Tp >, unordered_set< int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_set< _Tp, _Hash, _Pred, _Allocator >
- template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  vector (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> vector< _ValT, _Allocator >
- template<typename _Tp, typename _Allocator = allocator<_Tp>, typename = _RequireAllocator<_Allocator>>
  vector (size_t, _Tp, _Allocator=_Allocator()) -> vector< _Tp, _Allocator >

4.11.1 Detailed Description

GNU debug code, replaces standard behavior with debug behavior.

Macros and namespaces used by the implementation outside of debug wrappers to verify certain properties. The _↩ _glibcxx_requires_xxx macros are merely wrappers around the __glibcxx_check_xxx wrappers when we are compiling with debug mode, but disappear when we are in release mode so that there is no checking performed in, e.g., the standard library algorithms.

4.11.2 Function Documentation

swap()

```
template<typename _Tp, typename _Alloc>
void std::__debug::swap (
             forward_list< _Tp, _Alloc > & __lx,
             forward_list< _Tp, _Alloc > & __ly)  [inline], [noexcept]
```

See std::forward_list::swap().

4.12 std::__detail Namespace Reference

Classes

- struct _BracketMatcher
- class _Compiler
- class _Executor
- struct _List_node_base
- struct _List_node_header
- struct _Quoted_string
- class _Scanner
- class _StateSeq

Typedefs

- template<typename _Tp, typename _Up>
  using **__cmp3way_res_t**
- template<typename _Tp>
  using **__unsigned_least_t**
- template<typename _CharT>
  using **_Matcher**
- typedef long **_StateIdT**

Enumerations

- enum _Opcode : int {
  _S_opcode_unknown , **_S_opcode_alternative** , **_S_opcode_repeat** , **_S_opcode_backref** ,
  _S_opcode_line_begin_assertion , **_S_opcode_line_end_assertion** , **_S_opcode_word_boundary** , **_S_↩ opcode_subexpr_lookahead** ,
  _S_opcode_subexpr_begin , **_S_opcode_subexpr_end** , **_S_opcode_dummy** , **_S_opcode_match** ,
  _S_opcode_accept }
- enum class **_RegexExecutorPolicy** : int { **_S_auto** , **_S_alternate** }

Functions

- template<typename _Res, typename _Tp>
  constexpr _Res **__abs_r** (_Tp __val)
- template<typename>
  void **__abs_r** (bool)=delete

- template<typename _ChTraits>
  constexpr auto **__char_traits_cmp_cat** (int __cmp) noexcept
- template<typename... _Ts>
  constexpr auto **__common_cmp_cat** ()
- template<typename _Container, typename _UnsafeContainer, typename _Predicate>
  _Container::size_type **__erase_nodes_if** (_Container &__cont, _UnsafeContainer &__ucont, _Predicate __pred)
- template<typename _ValT, typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & **__extract_params** (basic_istream< _CharT, _Traits > &__is, vector< _ValT > &__vals, size_t __n)
- template<bool _DecOnly, typename _Tp>
  constexpr bool __from_chars_alnum (const char *&__first, const char *__last, _Tp &__val, int __base)
- template<bool _DecOnly = false>
  constexpr unsigned char **__from_chars_alnum_to_val** (unsigned char __c)
- template<bool _DecOnly, typename _Tp>
  constexpr bool __from_chars_pow2_base (const char *&__first, const char *__last, _Tp &__val, int __base)
- template<typename _Tp>
  constexpr _Tp **__gcd** (_Tp __m, _Tp __n)
- template<typename _Tp>
  _Tp * **__get_temporary_buffer** (ptrdiff_t __len) noexcept
- template<typename _Tp>
  constexpr bool **__p1_representable_as_double** (_Tp __x) noexcept
- template<typename _Tp>
  constexpr bool **__raise_and_add** (_Tp &__val, int __base, unsigned char __c)
- template<typename _BiIter, typename _Alloc, typename _CharT, typename _TraitsT>
  bool **__regex_algo_impl** (_BiIter __s, _BiIter __e, match_results< _BiIter, _Alloc > &__m, const basic_regex< _CharT, _TraitsT > &__re, regex_constants::match_flag_type __flags, _RegexExecutorPolicy __policy, bool __match_mode)
- template<typename _Tp>
  constexpr bool **__representable_as_double** (_Tp __x) noexcept
- template<typename _Tp>
  void **__return_temporary_buffer** (_Tp *__p, size_t __len)
- template<typename _Tp>
  constexpr to_chars_result **__to_chars** (char *__first, char *__last, _Tp __val, int __base) noexcept
- template<typename _Tp>
  constexpr to_chars_result **__to_chars_10** (char *__first, char *__last, _Tp __val) noexcept
- template<typename _Tp>
  constexpr void **__to_chars_10_impl** (char *__first, unsigned __len, _Tp __val) noexcept
- template<typename _Tp>
  constexpr to_chars_result **__to_chars_16** (char *__first, char *__last, _Tp __val) noexcept
- template<typename _Tp>
  constexpr to_chars_result **__to_chars_2** (char *__first, char *__last, _Tp __val) noexcept
- template<typename _Tp>
  constexpr to_chars_result **__to_chars_8** (char *__first, char *__last, _Tp __val) noexcept
- template<typename _Tp>
  constexpr unsigned **__to_chars_len** (_Tp __value, int __base) noexcept
- template<typename _Tp>
  constexpr unsigned **__to_chars_len_2** (_Tp __value) noexcept
- template<typename _CharT, typename _Traits, typename _String>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const _Quoted_string< _String, _CharT > &__str)
- template<typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const _Quoted_string< const _CharT *, _CharT > &__str)

- template<typename _CharT, typename _Traits, typename _Alloc>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, const _Quoted_string< basic_string< _CharT, _Traits, _Alloc > &, _CharT > &__str)

Variables

- template<typename _Tp>
  constexpr unsigned **__cmp_cat_id**
- template<> constexpr unsigned **__cmp_cat_id< partial_ordering >**
- template<> constexpr unsigned **__cmp_cat_id< strong_ordering >**
- template<> constexpr unsigned **__cmp_cat_id< weak_ordering >**
- constexpr _StateIdT **_S_invalid_state_id**

4.12.1 Detailed Description

Implementation details not part of the namespace std interface.

4.12.2 Function Documentation

__from_chars_alnum()

```
template<bool _DecOnly, typename _Tp>
bool std::__detail::__from_chars_alnum (
            const char *& __first,
            const char * __last,
            _Tp & __val,
            int __base)  [constexpr]
```

std::from_chars implementation for integers in any base. If _DecOnly is true, then we may assume __base is at most 10.

__from_chars_pow2_base()

```
template<bool _DecOnly, typename _Tp>
bool std::__detail::__from_chars_pow2_base (
            const char *& __first,
            const char * __last,
            _Tp & __val,
            int __base)  [constexpr]
```

std::from_chars implementation for integers in a power-of-two base. If _DecOnly is true, then we may assume __base is at most 8.

operator<<() [1/2]

```
template<typename _CharT, typename _Traits, typename _String>
std::basic_ostream< _CharT, _Traits > & std::__detail::operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const _Quoted_string< _String, _CharT > & __str)
```

Inserter for quoted strings.

operator<<() [2/2]

```
template<typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & std::__detail::operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const _Quoted_string< const _CharT *, _CharT > & __str)
```

Inserter for quoted strings.

operator>>()

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_istream< _CharT, _Traits > & std::__detail::operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            const _Quoted_string< basic_string< _CharT, _Traits, _Alloc > &, _CharT > & __str)
```

Extractor for delimited strings. The left and right delimiters can be different.
References std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::flags(), std::ios_base::fmtflags, std::basic_ios< _CharT, _Traits >::g
std::ios_base::setf(), std::ios_base::skipws, and std::basic_istream< _CharT, _Traits >::unget().

4.13 std::__parallel Namespace Reference

Classes

- struct _CRandNumber

Functions

- template<typename __RAIter, typename _Tp, typename _BinaryOperation>
  _Tp **__accumulate_switch** (__RAIter __begin, __RAIter __end, _Tp __init, _BinaryOperation __binary_op, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter, typename _Tp, typename _BinaryOperation, typename _IteratorTag>
  _Tp **__accumulate_switch** (_IIter __begin, _IIter __end, _Tp __init, _BinaryOperation __binary_op, _IteratorTag)
- template<typename _IIter, typename _Tp, typename _IteratorTag>
  _Tp **__accumulate_switch** (_IIter __begin, _IIter __end, _Tp __init, _IteratorTag)
- template<typename _IIter, typename _Tp, typename _BinaryOper, typename _Tag>
  _Tp **__accumulate_switch** (_IIter, _IIter, _Tp, _BinaryOper, _Tag)
- template<typename _IIter, typename _Tp, typename _Tag>
  _Tp **__accumulate_switch** (_IIter, _IIter, _Tp, _Tag)
- template<typename _RAIter, typename _Tp, typename _BinaryOper>
  _Tp **__accumulate_switch** (_RAIter, _RAIter, _Tp, _BinaryOper, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism=__gnu_parallel::parallel_unbalanced)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation, typename _IteratorTag1, typename _IteratorTag2>
  _OutputIterator **__adjacent_difference_switch** (_IIter __begin, _IIter __end, _OutputIterator __result, _Binary↩
  Operation __bin_op, _IteratorTag1, _IteratorTag2)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator **__adjacent_difference_switch** (_IIter __begin, _IIter __end, _OutputIterator __result, _Binary↩
  Operation __bin_op, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism ↩
  __parallelism_tag)
- template<typename _IIter, typename _OIter, typename _BinaryOper, typename _Tag1, typename _Tag2>
  _OIter **__adjacent_difference_switch** (_IIter, _IIter, _OIter, _BinaryOper, _Tag1, _Tag2)
- template<typename _IIter, typename _OIter, typename _BinaryOper>
  _OIter **__adjacent_difference_switch** (_IIter, _IIter, _OIter, _BinaryOper, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism=__gnu_parallel::parallel_unbalanced)
- template<typename _FIter, typename _BiPredicate, typename _IterTag>
  _FIter **__adjacent_find_switch** (_FIter, _FIter, _BiPredicate, _IterTag)
- template<typename _FIter, typename _IterTag>
  _FIter **__adjacent_find_switch** (_FIter, _FIter, _IterTag)
- template<typename _FIterator, typename _BinaryPredicate, typename _IteratorTag>
  _FIterator **__adjacent_find_switch** (_FIterator __begin, _FIterator __end, _BinaryPredicate __pred, _Iterator↩
  Tag)
- template<typename _FIterator, typename _IteratorTag>
  _FIterator **__adjacent_find_switch** (_FIterator __begin, _FIterator __end, _IteratorTag)

- template<typename _RAIter, typename _BinaryPredicate>
  _RAIter **__adjacent_find_switch** (_RAIter __begin, _RAIter __end, _BinaryPredicate __pred, random_access_iterator_tag)
- template<typename _RAIter>
  _RAIter **__adjacent_find_switch** (_RAIter __begin, _RAIter __end, random_access_iterator_tag)
- template<typename _RAIter, typename _BiPredicate>
  _RAIter **__adjacent_find_switch** (_RAIter, _RAIter, _BiPredicate, random_access_iterator_tag)
- template<typename _IIter, typename _Predicate, typename _IteratorTag>
  iterator_traits< _IIter >::difference_type **__count_if_switch** (_IIter __begin, _IIter __end, _Predicate __pred, _IteratorTag)
- template<typename _IIter, typename _Predicate, typename _IterTag>
  iterator_traits< _IIter >::difference_type **__count_if_switch** (_IIter, _IIter, _Predicate, _IterTag)
- template<typename _RAIter, typename _Predicate>
  iterator_traits< _RAIter >::difference_type **__count_if_switch** (_RAIter __begin, _RAIter __end, _Predicate _↩
  _pred, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter, typename _Tp, typename _IteratorTag>
  iterator_traits< _IIter >::difference_type **__count_switch** (_IIter __begin, _IIter __end, const _Tp &__value, _↩
  IteratorTag)
- template<typename _IIter, typename _Tp, typename _IterTag>
  iterator_traits< _IIter >::difference_type **__count_switch** (_IIter, _IIter, const _Tp &, _IterTag)
- template<typename _RAIter, typename _Tp>
  iterator_traits< _RAIter >::difference_type **__count_switch** (_RAIter __begin, _RAIter __end, const _Tp &__↩
  value, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2>
  bool **__equal_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Predicate __pred, _IteratorTag1, _IteratorTag2)
- template<typename _RAIter1, typename _RAIter2, typename _Predicate>
  bool **__equal_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter, typename _FIterator, typename _BinaryPredicate, typename _IteratorTag1, typename _IteratorTag2>
  _IIter **__find_first_of_switch** (_IIter __begin1, _IIter __end1, _FIterator __begin2, _FIterator __end2, _Binary↩
  Predicate __comp, _IteratorTag1, _IteratorTag2)
- template<typename _IIter, typename _FIterator, typename _IteratorTag1, typename _IteratorTag2>
  _IIter **__find_first_of_switch** (_IIter __begin1, _IIter __end1, _FIterator __begin2, _FIterator __end2, _Iterator↩
  Tag1, _IteratorTag2)
- template<typename _IIter, typename _FIter, typename _BiPredicate, typename _IterTag1, typename _IterTag2>
  _IIter **__find_first_of_switch** (_IIter, _IIter, _FIter, _FIter, _BiPredicate, _IterTag1, _IterTag2)
- template<typename _IIter, typename _FIter, typename _IterTag1, typename _IterTag2>
  _IIter **__find_first_of_switch** (_IIter, _IIter, _FIter, _FIter, _IterTag1, _IterTag2)
- template<typename _RAIter, typename _FIterator, typename _BinaryPredicate, typename _IteratorTag>
  _RAIter **__find_first_of_switch** (_RAIter __begin1, _RAIter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, random_access_iterator_tag, _IteratorTag)
- template<typename _RAIter, typename _FIter, typename _BiPredicate, typename _IterTag>
  _RAIter **__find_first_of_switch** (_RAIter, _RAIter, _FIter, _FIter, _BiPredicate, random_access_iterator_tag, _↩
  IterTag)
- template<typename _IIter, typename _Predicate, typename _IteratorTag>
  _IIter **__find_if_switch** (_IIter __begin, _IIter __end, _Predicate __pred, _IteratorTag)
- template<typename _IIter, typename _Predicate, typename _IterTag>
  _IIter **__find_if_switch** (_IIter, _IIter, _Predicate, _IterTag)
- template<typename _RAIter, typename _Predicate>
  _RAIter **__find_if_switch** (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)
- template<typename _IIter, typename _Tp, typename _IteratorTag>
  _IIter **__find_switch** (_IIter __begin, _IIter __end, const _Tp &__val, _IteratorTag)

- template<typename _IIter, typename _Tp, typename _IterTag>
  _IIter **__find_switch** (_IIter, _IIter, const _Tp &, _IterTag)
- template<typename _RAIter, typename _Tp>
  _RAIter **__find_switch** (_RAIter __begin, _RAIter __end, const _Tp &__val, random_access_iterator_tag)
- template<typename _IIter, typename _Function, typename _IteratorTag>
  _Function **__for_each_switch** (_IIter __begin, _IIter __end, _Function __f, _IteratorTag)
- template<typename _IIter, typename _Function, typename _IterTag>
  _Function **__for_each_switch** (_IIter, _IIter, _Function, _IterTag)
- template<typename _RAIter, typename _Function>
  _Function **__for_each_switch** (_RAIter __begin, _RAIter __end, _Function __f, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _OIter, typename _Size, typename _Generator, typename _IterTag>
  _OIter **__generate_n_switch** (_OIter, _Size, _Generator, _IterTag)
- template<typename _OutputIterator, typename _Size, typename _Generator, typename _IteratorTag>
  _OutputIterator **__generate_n_switch** (_OutputIterator __begin, _Size __n, _Generator __gen, _IteratorTag)
- template<typename _RAIter, typename _Size, typename _Generator>
  _RAIter **__generate_n_switch** (_RAIter __begin, _Size __n, _Generator __gen, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIter, typename _Generator, typename _IterTag>
  void **__generate_switch** (_FIter, _FIter, _Generator, _IterTag)
- template<typename _FIterator, typename _Generator, typename _IteratorTag>
  void **__generate_switch** (_FIterator __begin, _FIterator __end, _Generator __gen, _IteratorTag)
- template<typename _RAIter, typename _Generator>
  void **__generate_switch** (_RAIter __begin, _RAIter __end, _Generator __gen, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter1, typename _IIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2, typename _IteratorTag1, typename _IteratorTag2>
  _Tp **__inner_product_switch** (_IIter1 __first1, _IIter1 __last1, _IIter2 __first2, _Tp __init, _BinaryFunction1 _binary_op1, _BinaryFunction2 __binary_op2, _IteratorTag1, _IteratorTag2)
- template<typename _IIter1, typename _IIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2, typename _Tag1, typename _Tag2>
  _Tp **__inner_product_switch** (_IIter1, _IIter1, _IIter2, _Tp, _BinaryFunction1, _BinaryFunction2, _Tag1, _Tag2)
- template<typename _RAIter1, typename _RAIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>
  _Tp **__inner_product_switch** (_RAIter1, _RAIter1, _RAIter2, _Tp, _BinaryFunction1, _BinaryFunction2, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism=__gnu_parallel::parallel_unbalanced)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2>
  bool **__lexicographical_compare_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Predicate __pred, _IteratorTag1, _IteratorTag2)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _IterTag1, typename _IterTag2>
  bool **__lexicographical_compare_switch** (_IIter1, _IIter1, _IIter2, _IIter2, _Predicate, _IterTag1, _IterTag2)
- template<typename _RAIter1, typename _RAIter2, typename _Predicate>
  bool **__lexicographical_compare_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _FIter, typename _Compare, typename _IterTag>
  _FIter **__max_element_switch** (_FIter, _FIter, _Compare, _IterTag)
- template<typename _FIterator, typename _Compare, typename _IteratorTag>
  _FIterator **__max_element_switch** (_FIterator __begin, _FIterator __end, _Compare __comp, _IteratorTag)
- template<typename _RAIter, typename _Compare>
  _RAIter **__max_element_switch** (_RAIter __begin, _RAIter __end, _Compare __comp, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Compare, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3>
  _OutputIterator **__merge_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Compare __comp, _IteratorTag1, _IteratorTag2, _IteratorTag3)

- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Compare>
  _OutputIterator **__merge_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, ↩
  _OutputIterator __result, _Compare __comp, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare, typename _IterTag1, typename _IterTag2, typename _IterTag3>
  _OIter **__merge_switch** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare, _IterTag1, _IterTag2, _IterTag3)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare>
  _OIter **__merge_switch** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _FIter, typename _Compare, typename _IterTag>
  _FIter **__min_element_switch** (_FIter, _FIter, _Compare, _IterTag)
- template<typename _FIterator, typename _Compare, typename _IteratorTag>
  _FIterator **__min_element_switch** (_FIterator __begin, _FIterator __end, _Compare __comp, _IteratorTag)
- template<typename _RAIter, typename _Compare>
  _RAIter **__min_element_switch** (_RAIter __begin, _RAIter __end, _Compare __comp, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2>
  pair< _IIter1, _IIter2 > **__mismatch_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Predicate __pred, _IteratorTag1, _IteratorTag2)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2>
  pair< _IIter1, _IIter2 > **__mismatch_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate _↩
  _pred, _IteratorTag1, _IteratorTag2)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _IterTag1, typename _IterTag2>
  pair< _IIter1, _IIter2 > **__mismatch_switch** (_IIter1, _IIter1, _IIter2, _Predicate, _IterTag1, _IterTag2)
- template<typename _RAIter1, typename _RAIter2, typename _Predicate>
  pair< _RAIter1, _RAIter2 > **__mismatch_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _RAIter1, typename _RAIter2, typename _Predicate>
  pair< _RAIter1, _RAIter2 > **__mismatch_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation, typename _IteratorTag1, typename _IteratorTag2>
  _OutputIterator **__partial_sum_switch** (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __bin_op, _IteratorTag1, _IteratorTag2)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator **__partial_sum_switch** (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __bin_op, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter, typename _OIter, typename _BinaryOper, typename _Tag1, typename _Tag2>
  _OIter **__partial_sum_switch** (_IIter, _IIter, _OIter, _BinaryOper, _Tag1, _Tag2)
- template<typename _IIter, typename _OIter, typename _BinaryOper>
  _OIter **__partial_sum_switch** (_IIter, _IIter, _OIter, _BinaryOper, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _FIter, typename _Predicate, typename _IterTag>
  _FIter **__partition_switch** (_FIter, _FIter, _Predicate, _IterTag)
- template<typename _FIterator, typename _Predicate, typename _IteratorTag>
  _FIterator **__partition_switch** (_FIterator __begin, _FIterator __end, _Predicate __pred, _IteratorTag)
- template<typename _RAIter, typename _Predicate>
  _RAIter **__partition_switch** (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)
- template<typename _FIter, typename _Predicate, typename _Tp, typename _IterTag>
  void **__replace_if_switch** (_FIter, _FIter, _Predicate, const _Tp &, _IterTag)
- template<typename _FIterator, typename _Predicate, typename _Tp, typename _IteratorTag>
  void **__replace_if_switch** (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value, _IteratorTag)

- template<typename _RAIter, typename _Predicate, typename _Tp>
  void **__replace_if_switch** (_RAIter __begin, _RAIter __end, _Predicate __pred, const _Tp &__new_value, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIter, typename _Tp, typename _IterTag>
  void **__replace_switch** (_FIter, _FIter, const _Tp &, const _Tp &, _IterTag)
- template<typename _FIterator, typename _Tp, typename _IteratorTag>
  void **__replace_switch** (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value, _IteratorTag)
- template<typename _RAIter, typename _Tp>
  void **__replace_switch** (_RAIter __begin, _RAIter __end, const _Tp &__old_value, const _Tp &__new_value, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIter, typename _Integer, typename _Tp, typename _BiPredicate, typename _IterTag>
  _FIter **__search_n_switch** (_FIter, _FIter, _Integer, const _Tp &, _BiPredicate, _IterTag)
- template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate, typename _IteratorTag>
  _FIterator **__search_n_switch** (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, _IteratorTag)
- template<typename _RAIter, typename _Integer, typename _Tp, typename _BinaryPredicate>
  _RAIter **__search_n_switch** (_RAIter __begin, _RAIter __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, random_access_iterator_tag)
- template<typename _RAIter, typename _Integer, typename _Tp, typename _BiPredicate>
  _RAIter **__search_n_switch** (_RAIter, _RAIter, _Integer, const _Tp &, _BiPredicate, random_access_iterator_tag)
- template<typename _FIter1, typename _FIter2, typename _BiPredicate, typename _IterTag1, typename _IterTag2>
  _FIter1 **__search_switch** (_FIter1, _FIter1, _FIter2, _FIter2, _BiPredicate, _IterTag1, _IterTag2)
- template<typename _FIter1, typename _FIter2, typename _IterTag1, typename _IterTag2>
  _FIter1 **__search_switch** (_FIter1, _FIter1, _FIter2, _FIter2, _IterTag1, _IterTag2)
- template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate, typename _IteratorTag1, typename _IteratorTag2>
  _FIterator1 **__search_switch** (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred, _IteratorTag1, _IteratorTag2)
- template<typename _FIterator1, typename _FIterator2, typename _IteratorTag1, typename _IteratorTag2>
  _FIterator1 **__search_switch** (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _IteratorTag1, _IteratorTag2)
- template<typename _RAIter1, typename _RAIter2, typename _BinaryPredicate>
  _RAIter1 **__search_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _BinaryPredicate __pred, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _RAIter1, typename _RAIter2>
  _RAIter1 **__search_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _RAIter1, typename _RAIter2, typename _BiPredicate>
  _RAIter1 **__search_switch** (_RAIter1, _RAIter1, _RAIter2, _RAIter2, _BiPredicate, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3>
  _OutputIterator **__set_difference_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3>
  _OIter **__set_difference_switch** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)
- template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate>
  _Output_RAIter **__set_difference_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)

- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _↩ IteratorTag2, typename _IteratorTag3>
  _OutputIterator **__set_intersection_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3>
  _OIter **__set_intersection_switch** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, _IterTag1, _IterTag2, _↩ IterTag3)
- template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate>
  _Output_RAIter **__set_intersection_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _↩ IteratorTag2, typename _IteratorTag3>
  _OutputIterator **__set_symmetric_difference_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3>
  _OIter **__set_symmetric_difference_switch** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, _IterTag1, _↩ IterTag2, _IterTag3)
- template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate>
  _Output_RAIter **__set_symmetric_difference_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 ↩ __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _↩ IteratorTag2, typename _IteratorTag3>
  _OutputIterator **__set_union_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _↩ OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3>
  _OIter **__set_union_switch** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)
- template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate>
  _Output_RAIter **__set_union_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 _↩ _end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter, typename _OIter, typename _UnaryOperation, typename _IterTag1, typename _IterTag2>
  _OIter **__transform1_switch** (_IIter, _IIter, _OIter, _UnaryOperation, _IterTag1, _IterTag2)
- template<typename _RAIIter, typename _RAOIter, typename _UnaryOperation>
  _RAOIter **__transform1_switch** (_RAIIter, _RAIIter, _RAOIter, _UnaryOperation, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism=__gnu_parallel::parallel_balanced)
- template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation, typename _IteratorTag1, typename _IteratorTag2>
  _RAIter2 **__transform1_switch** (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __↩ unary_op, _IteratorTag1, _IteratorTag2)
- template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation>
  _RAIter2 **__transform1_switch** (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __unary_op, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism __↩ parallelism_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _BinaryOperation, typename _Tag1, typename _Tag2, typename _Tag3>
  _OutputIterator **__transform2_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _OutputIterator __↩ result, _BinaryOperation __binary_op, _Tag1, _Tag2, _Tag3)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _BiOperation, typename _Tag1, typename _Tag2, typename _↩ Tag3>
  _OIter **__transform2_switch** (_IIter1, _IIter1, _IIter2, _OIter, _BiOperation, _Tag1, _Tag2, _Tag3)

- template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BinaryOperation>
  _RAIter3 **__transform2_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter3 __result, ↩
  _BinaryOperation __binary_op, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BiOperation>
  _RAIter3 **__transform2_switch** (_RAIter1, _RAIter1, _RAIter2, _RAIter3, _BiOperation, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism=__gnu_parallel::parallel_bal
- template<typename _IIter, typename _OutputIterator, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2>
  _OutputIterator **__unique_copy_switch** (_IIter __begin, _IIter __last, _OutputIterator __out, _Predicate __pred, _IteratorTag1, _IteratorTag2)
- template<typename _IIter, typename _OIter, typename _Predicate, typename _IterTag1, typename _IterTag2>
  _OIter **__unique_copy_switch** (_IIter, _IIter, _OIter, _Predicate, _IterTag1, _IterTag2)
- template<typename _RAIter, typename _RandomAccessOutputIterator, typename _Predicate>
  _RandomAccessOutputIterator **__unique_copy_switch** (_RAIter __begin, _RAIter __last, _RandomAccess↩
  OutputIterator __out, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _RAIter, typename _RandomAccess_OIter, typename _Predicate>
  _RandomAccess_OIter **__unique_copy_switch** (_RAIter, _RAIter, _RandomAccess_OIter, _Predicate, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter, typename _Tp, typename _BinaryOperation>
  _Tp **accumulate** (_IIter __begin, _IIter __end, _Tp __init, _BinaryOperation __binary_op)
- template<typename _IIter, typename _Tp, typename _BinaryOperation>
  _Tp **accumulate** (_IIter __begin, _IIter __end, _Tp __init, _BinaryOperation __binary_op, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter, typename _Tp, typename _BinaryOperation>
  _Tp **accumulate** (_IIter __begin, _IIter __end, _Tp __init, _BinaryOperation __binary_op, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _Tp>
  _Tp **accumulate** (_IIter, _IIter, _Tp)
- template<typename _IIter, typename _Tp>
  _Tp **accumulate** (_IIter, _IIter, _Tp, __gnu_parallel::_Parallelism)
- template<typename _IIter, typename _Tp>
  _Tp **accumulate** (_IIter, _IIter, _Tp, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _Tp, typename _BinaryOper>
  _Tp **accumulate** (_IIter, _IIter, _Tp, _BinaryOper)
- template<typename _IIter, typename _Tp, typename _BinaryOper>
  _Tp **accumulate** (_IIter, _IIter, _Tp, _BinaryOper, __gnu_parallel::_Parallelism)
- template<typename _IIter, typename _Tp, typename _BinaryOper>
  _Tp **accumulate** (_IIter, _IIter, _Tp, _BinaryOper, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OutputIterator>
  _OutputIterator **adjacent_difference** (_IIter __begin, _IIter __end, _OutputIterator __result)
- template<typename _IIter, typename _OutputIterator>
  _OutputIterator **adjacent_difference** (_IIter __begin, _IIter __end, _OutputIterator __result, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter, typename _OutputIterator>
  _OutputIterator **adjacent_difference** (_IIter __begin, _IIter __end, _OutputIterator __result, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator **adjacent_difference** (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __bin_op, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator **adjacent_difference** (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __binary_op)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator **adjacent_difference** (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __binary_op, __gnu_parallel::_Parallelism __parallelism_tag)

- template<typename _IIter, typename _OIter>
  _OIter **adjacent_difference** (_IIter, _IIter, _OIter)
- template<typename _IIter, typename _OIter>
  _OIter **adjacent_difference** (_IIter, _IIter, _OIter, __gnu_parallel::_Parallelism)
- template<typename _IIter, typename _OIter>
  _OIter **adjacent_difference** (_IIter, _IIter, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OIter, typename _BinaryOper>
  _OIter **adjacent_difference** (_IIter, _IIter, _OIter, _BinaryOper)
- template<typename _IIter, typename _OIter, typename _BinaryOper>
  _OIter **adjacent_difference** (_IIter, _IIter, _OIter, _BinaryOper, __gnu_parallel::_Parallelism)
- template<typename _IIter, typename _OIter, typename _BinaryOper>
  _OIter **adjacent_difference** (_IIter, _IIter, _OIter, _BinaryOper, __gnu_parallel::sequential_tag)
- template<typename _FIter>
  _FIter **adjacent_find** (_FIter, _FIter)
- template<typename _FIter>
  _FIter **adjacent_find** (_FIter, _FIter, __gnu_parallel::sequential_tag)
- template<typename _FIter, typename _BiPredicate>
  _FIter **adjacent_find** (_FIter, _FIter, _BiPredicate)
- template<typename _FIter, typename _BiPredicate>
  _FIter **adjacent_find** (_FIter, _FIter, _BiPredicate, __gnu_parallel::sequential_tag)
- template<typename _FIterator>
  _FIterator **adjacent_find** (_FIterator __begin, _FIterator __end)
- template<typename _FIterator>
  _FIterator **adjacent_find** (_FIterator __begin, _FIterator __end, __gnu_parallel::sequential_tag)
- template<typename _FIterator, typename _BinaryPredicate>
  _FIterator **adjacent_find** (_FIterator __begin, _FIterator __end, _BinaryPredicate __binary_pred, __gnu_parallel::sequential_tag)
- template<typename _FIterator, typename _BinaryPredicate>
  _FIterator **adjacent_find** (_FIterator __begin, _FIterator __end, _BinaryPredicate __pred)
- template<typename _IIter, typename _Tp>
  iterator_traits< _IIter >::difference_type **count** (_IIter __begin, _IIter __end, const _Tp &__value)
- template<typename _IIter, typename _Tp>
  iterator_traits< _IIter >::difference_type **count** (_IIter __begin, _IIter __end, const _Tp &__value, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter, typename _Tp>
  iterator_traits< _IIter >::difference_type **count** (_IIter __begin, _IIter __end, const _Tp &__value, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _Predicate>
  iterator_traits< _IIter >::difference_type **count_if** (_IIter __begin, _IIter __end, _Predicate __pred)
- template<typename _IIter, typename _Predicate>
  iterator_traits< _IIter >::difference_type **count_if** (_IIter __begin, _IIter __end, _Predicate __pred, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter, typename _Predicate>
  iterator_traits< _IIter >::difference_type **count_if** (_IIter __begin, _IIter __end, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2>
  constexpr bool **equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2)
- template<typename _IIter1, typename _IIter2>
  bool **equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2>
  constexpr bool **equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2)
- template<typename _IIter1, typename _IIter2>
  bool **equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, __gnu_parallel::sequential_tag)

- template<typename _IIter1, typename _IIter2, typename _BinaryPredicate>
  constexpr bool **equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _BinaryPredicate __binary_pred)
- template<typename _IIter1, typename _IIter2, typename _BinaryPredicate>
  bool **equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _BinaryPredicate __binary_pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate>
  constexpr bool **equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred)
- template<typename _IIter1, typename _IIter2, typename _Predicate>
  bool **equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _Tp>
  _IIter **find** (_IIter __begin, _IIter __end, const _Tp &__val)
- template<typename _IIter, typename _Tp>
  _IIter **find** (_IIter __begin, _IIter __end, const _Tp &__val, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _FIterator>
  _IIter **find_first_of** (_IIter __begin1, _IIter __end1, _FIterator __begin2, _FIterator __end2)
- template<typename _IIter, typename _FIterator>
  _IIter **find_first_of** (_IIter __begin1, _IIter __end1, _FIterator __begin2, _FIterator __end2, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _FIterator, typename _BinaryPredicate>
  _IIter **find_first_of** (_IIter __begin1, _IIter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp)
- template<typename _IIter, typename _FIterator, typename _BinaryPredicate>
  _IIter **find_first_of** (_IIter __begin1, _IIter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _FIter>
  _IIter **find_first_of** (_IIter, _IIter, _FIter, _FIter)
- template<typename _IIter, typename _FIter>
  _IIter **find_first_of** (_IIter, _IIter, _FIter, _FIter, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _FIter, typename _BiPredicate>
  _IIter **find_first_of** (_IIter, _IIter, _FIter, _FIter, _BiPredicate)
- template<typename _IIter, typename _FIter, typename _BiPredicate>
  _IIter **find_first_of** (_IIter, _IIter, _FIter, _FIter, _BiPredicate, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _Predicate>
  _IIter **find_if** (_IIter __begin, _IIter __end, _Predicate __pred)
- template<typename _IIter, typename _Predicate>
  _IIter **find_if** (_IIter __begin, _IIter __end, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _Function>
  _Function **for_each** (_IIter __begin, _IIter __end, _Function __f, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _Function>
  _Function **for_each** (_IIter, _IIter, _Function)
- template<typename _Iterator, typename _Function>
  _Function **for_each** (_Iterator __begin, _Iterator __end, _Function __f)
- template<typename _Iterator, typename _Function>
  _Function **for_each** (_Iterator __begin, _Iterator __end, _Function __f, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIter, typename _Generator>
  void **generate** (_FIter, _FIter, _Generator)
- template<typename _FIter, typename _Generator>
  void **generate** (_FIter, _FIter, _Generator, __gnu_parallel::_Parallelism)
- template<typename _FIter, typename _Generator>
  void **generate** (_FIter, _FIter, _Generator, __gnu_parallel::sequential_tag)
- template<typename _FIterator, typename _Generator>
  void **generate** (_FIterator __begin, _FIterator __end, _Generator __gen)

- template<typename _FIterator, typename _Generator>
  void **generate** (_FIterator __begin, _FIterator __end, _Generator __gen, __gnu_parallel::_Parallelism __↩
  parallelism_tag)
- template<typename _FIterator, typename _Generator>
  void **generate** (_FIterator __begin, _FIterator __end, _Generator __gen, __gnu_parallel::sequential_tag)
- template<typename _OIter, typename _Size, typename _Generator>
  _OIter **generate_n** (_OIter, _Size, _Generator)
- template<typename _OIter, typename _Size, typename _Generator>
  _OIter **generate_n** (_OIter, _Size, _Generator, __gnu_parallel::_Parallelism)
- template<typename _OIter, typename _Size, typename _Generator>
  _OIter **generate_n** (_OIter, _Size, _Generator, __gnu_parallel::sequential_tag)
- template<typename _OutputIterator, typename _Size, typename _Generator>
  _OutputIterator **generate_n** (_OutputIterator __begin, _Size __n, _Generator __gen)
- template<typename _OutputIterator, typename _Size, typename _Generator>
  _OutputIterator **generate_n** (_OutputIterator __begin, _Size __n, _Generator __gen, __gnu_parallel::_Parallelism
  __parallelism_tag)
- template<typename _OutputIterator, typename _Size, typename _Generator>
  _OutputIterator **generate_n** (_OutputIterator __begin, _Size __n, _Generator __gen, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _Tp>
  _Tp **inner_product** (_IIter1, _IIter1, _IIter2, _Tp)
- template<typename _IIter1, typename _IIter2, typename _Tp>
  _Tp **inner_product** (_IIter1, _IIter1, _IIter2, _Tp, __gnu_parallel::_Parallelism)
- template<typename _IIter1, typename _IIter2, typename _Tp>
  _Tp **inner_product** (_IIter1, _IIter1, _IIter2, _Tp, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>
  _Tp **inner_product** (_IIter1, _IIter1, _IIter2, _Tp, _BinaryFunction1, _BinaryFunction2)
- template<typename _IIter1, typename _IIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>
  _Tp **inner_product** (_IIter1, _IIter1, _IIter2, _Tp, _BinaryFunction1, _BinaryFunction2, __gnu_parallel::_Parallelism)
- template<typename _IIter1, typename _IIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>
  _Tp **inner_product** (_IIter1, _IIter1, _IIter2, _Tp, _BinaryFunction1, _BinaryFunction2, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2>
  constexpr bool **lexicographical_compare** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2)
- template<typename _IIter1, typename _IIter2>
  bool **lexicographical_compare** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2,
  __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate>
  constexpr bool **lexicographical_compare** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2,
  _Predicate __pred)
- template<typename _IIter1, typename _IIter2, typename _Predicate>
  bool **lexicographical_compare** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Predicate
  __pred, __gnu_parallel::sequential_tag)
- template<typename _FIter>
  _FIter **max_element** (_FIter, _FIter)
- template<typename _FIter>
  _FIter **max_element** (_FIter, _FIter, __gnu_parallel::_Parallelism)
- template<typename _FIter>
  _FIter **max_element** (_FIter, _FIter, __gnu_parallel::sequential_tag)
- template<typename _FIter, typename _Compare>
  _FIter **max_element** (_FIter, _FIter, _Compare)
- template<typename _FIter, typename _Compare>
  _FIter **max_element** (_FIter, _FIter, _Compare, __gnu_parallel::_Parallelism)
- template<typename _FIter, typename _Compare>
  _FIter **max_element** (_FIter, _FIter, _Compare, __gnu_parallel::sequential_tag)

- template<typename _FIterator>
  _FIterator **max_element** (_FIterator __begin, _FIterator __end)
- template<typename _FIterator>
  _FIterator **max_element** (_FIterator __begin, _FIterator __end, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator>
  _FIterator **max_element** (_FIterator __begin, _FIterator __end, __gnu_parallel::sequential_tag)
- template<typename _FIterator, typename _Compare>
  _FIterator **max_element** (_FIterator __begin, _FIterator __end, _Compare __comp)
- template<typename _FIterator, typename _Compare>
  _FIterator **max_element** (_FIterator __begin, _FIterator __end, _Compare __comp, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator, typename _Compare>
  _FIterator **max_element** (_FIterator __begin, _FIterator __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator>
  _OutputIterator **merge** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator>
  _OutputIterator **merge** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Compare>
  _OutputIterator **merge** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Compare __comp)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Compare>
  _OutputIterator **merge** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  _OIter **merge** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  _OIter **merge** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare>
  _OIter **merge** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare>
  _OIter **merge** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare, __gnu_parallel::sequential_tag)
- template<typename _FIter>
  _FIter **min_element** (_FIter, _FIter)
- template<typename _FIter>
  _FIter **min_element** (_FIter, _FIter, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIter>
  _FIter **min_element** (_FIter, _FIter, __gnu_parallel::sequential_tag)
- template<typename _FIter, typename _Compare>
  _FIter **min_element** (_FIter, _FIter, _Compare)
- template<typename _FIter, typename _Compare>
  _FIter **min_element** (_FIter, _FIter, _Compare, __gnu_parallel::_Parallelism)
- template<typename _FIter, typename _Compare>
  _FIter **min_element** (_FIter, _FIter, _Compare, __gnu_parallel::sequential_tag)
- template<typename _FIterator>
  _FIterator **min_element** (_FIterator __begin, _FIterator __end)
- template<typename _FIterator>
  _FIterator **min_element** (_FIterator __begin, _FIterator __end, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator>
  _FIterator **min_element** (_FIterator __begin, _FIterator __end, __gnu_parallel::sequential_tag)
- template<typename _FIterator, typename _Compare>
  _FIterator **min_element** (_FIterator __begin, _FIterator __end, _Compare __comp)

- template<typename _FIterator, typename _Compare>
  _FIterator **min_element** (_FIterator __begin, _FIterator __end, _Compare __comp, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator, typename _Compare>
  _FIterator **min_element** (_FIterator __begin, _FIterator __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2>
  pair< _IIter1, _IIter2 > **mismatch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2)
- template<typename _IIter1, typename _IIter2>
  pair< _IIter1, _IIter2 > **mismatch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2>
  pair< _IIter1, _IIter2 > **mismatch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2)
- template<typename _IIter1, typename _IIter2, typename _Predicate>
  pair< _IIter1, _IIter2 > **mismatch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred)
- template<typename _IIter1, typename _IIter2, typename _Predicate>
  pair< _IIter1, _IIter2 > **mismatch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>
  pair< _InputIterator1, _InputIterator2 > **mismatch** (_InputIterator1 __begin1, _InputIterator1 __end1, _InputIterator2 __begin2, _InputIterator2 __end2, _BinaryPredicate __binary_pred)
- template<typename _InputIterator1, typename _InputIterator2>
  pair< _InputIterator1, _InputIterator2 > **mismatch** (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, __gnu_parallel::sequential_tag)
- template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>
  pair< _InputIterator1, _InputIterator2 > **mismatch** (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _BinaryPredicate __binary_pred, __gnu_parallel::sequential_tag)
- template<typename _RAIter>
  void **nth_element** (_RAIter __begin, _RAIter __nth, _RAIter __end)
- template<typename _RAIter>
  void **nth_element** (_RAIter __begin, _RAIter __nth, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter, typename _Compare>
  void **nth_element** (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)
- template<typename _RAIter, typename _Compare>
  void **nth_element** (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIter>
  void **partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end)
- template<typename _RAIter>
  void **partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter, typename _Compare>
  void **partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)
- template<typename _RAIter, typename _Compare>
  void **partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OutputIterator>
  _OutputIterator **partial_sum** (_IIter __begin, _IIter __end, _OutputIterator __result)
- template<typename _IIter, typename _OutputIterator>
  _OutputIterator **partial_sum** (_IIter __begin, _IIter __end, _OutputIterator __result, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator **partial_sum** (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __bin_op, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator **partial_sum** (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __binary_op)
- template<typename _IIter, typename _OIter>
  _OIter **partial_sum** (_IIter, _IIter, _OIter __result)

- template<typename _IIter, typename _OIter>
  _OIter **partial_sum** (_IIter, _IIter, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OIter, typename _BinaryOper>
  _OIter **partial_sum** (_IIter, _IIter, _OIter, _BinaryOper)
- template<typename _IIter, typename _OIter, typename _BinaryOper>
  _OIter **partial_sum** (_IIter, _IIter, _OIter, _BinaryOper, __gnu_parallel::sequential_tag)
- template<typename _FIter, typename _Predicate>
  _FIter **partition** (_FIter, _FIter, _Predicate)
- template<typename _FIter, typename _Predicate>
  _FIter **partition** (_FIter, _FIter, _Predicate, __gnu_parallel::sequential_tag)
- template<typename _FIterator, typename _Predicate>
  _FIterator **partition** (_FIterator __begin, _FIterator __end, _Predicate __pred)
- template<typename _FIterator, typename _Predicate>
  _FIterator **partition** (_FIterator __begin, _FIterator __end, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _RAIter>
  void **random_shuffle** (_RAIter __begin, _RAIter __end)
- template<typename _RAIter>
  void **random_shuffle** (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter, typename _RandomNumberGenerator>
  void **random_shuffle** (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &&__rand)
- template<typename _RAIter, typename _RandomNumberGenerator>
  void **random_shuffle** (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &__rand, __gnu_parallel::sequential_tag)
- template<typename _FIter, typename _Tp>
  void **replace** (_FIter, _FIter, const _Tp &, const _Tp &)
- template<typename _FIter, typename _Tp>
  void **replace** (_FIter, _FIter, const _Tp &, const _Tp &, __gnu_parallel::_Parallelism)
- template<typename _FIter, typename _Tp>
  void **replace** (_FIter, _FIter, const _Tp &, const _Tp &, __gnu_parallel::sequential_tag)
- template<typename _FIterator, typename _Tp>
  void **replace** (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value)
- template<typename _FIterator, typename _Tp>
  void **replace** (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator, typename _Tp>
  void **replace** (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value, __gnu_parallel::sequential_tag)
- template<typename _FIter, typename _Predicate, typename _Tp>
  void **replace_if** (_FIter, _FIter, _Predicate, const _Tp &)
- template<typename _FIter, typename _Predicate, typename _Tp>
  void **replace_if** (_FIter, _FIter, _Predicate, const _Tp &, __gnu_parallel::_Parallelism)
- template<typename _FIter, typename _Predicate, typename _Tp>
  void **replace_if** (_FIter, _FIter, _Predicate, const _Tp &, __gnu_parallel::sequential_tag)
- template<typename _FIterator, typename _Predicate, typename _Tp>
  void **replace_if** (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value)
- template<typename _FIterator, typename _Predicate, typename _Tp>
  void **replace_if** (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator, typename _Predicate, typename _Tp>
  void **replace_if** (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value, __gnu_parallel::sequential_tag)
- template<typename _FIter1, typename _FIter2>
  _FIter1 **search** (_FIter1, _FIter1, _FIter2, _FIter2)

- template<typename _FIter1, typename _FIter2>
  _FIter1 **search** (_FIter1, _FIter1, _FIter2, _FIter2, __gnu_parallel::sequential_tag)
- template<typename _FIter1, typename _FIter2, typename _BiPredicate>
  constexpr _FIter1 **search** (_FIter1, _FIter1, _FIter2, _FIter2, _BiPredicate)
- template<typename _FIter1, typename _FIter2, typename _BiPredicate>
  _FIter1 **search** (_FIter1, _FIter1, _FIter2, _FIter2, _BiPredicate, __gnu_parallel::sequential_tag)
- template<typename _FIterator1, typename _FIterator2>
  _FIterator1 **search** (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2)
- template<typename _FIterator1, typename _FIterator2>
  _FIterator1 **search** (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, __gnu_parallel::sequential_tag)
- template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate>
  constexpr _FIterator1 **search** (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred)
- template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate>
  _FIterator1 **search** (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred, __gnu_parallel::sequential_tag)
- template<typename _ForwardIterator, typename _Searcher>
  _ForwardIterator search (_ForwardIterator __first, _ForwardIterator __last, const _Searcher &__searcher)
- template<typename _FIter, typename _Integer, typename _Tp>
  _FIter **search_n** (_FIter, _FIter, _Integer, const _Tp &)
- template<typename _FIter, typename _Integer, typename _Tp>
  _FIter **search_n** (_FIter, _FIter, _Integer, const _Tp &, __gnu_parallel::sequential_tag)
- template<typename _FIter, typename _Integer, typename _Tp, typename _BiPredicate>
  _FIter **search_n** (_FIter, _FIter, _Integer, const _Tp &, _BiPredicate)
- template<typename _FIter, typename _Integer, typename _Tp, typename _BiPredicate>
  _FIter **search_n** (_FIter, _FIter, _Integer, const _Tp &, _BiPredicate, __gnu_parallel::sequential_tag)
- template<typename _FIterator, typename _Integer, typename _Tp>
  _FIterator **search_n** (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val)
- template<typename _FIterator, typename _Integer, typename _Tp>
  _FIterator **search_n** (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, __gnu_parallel::sequential_tag)
- template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate>
  _FIterator **search_n** (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)
- template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate>
  _FIterator **search_n** (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator>
  _OutputIterator **set_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator>
  _OutputIterator **set_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate>
  _OutputIterator **set_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate>
  _OutputIterator **set_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  _OIter **set_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  _OIter **set_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, __gnu_parallel::sequential_tag)

- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate>
  _OIter **set_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate>
  _OIter **set_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator>
  _OutputIterator **set_intersection** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Output↩
  Iterator __out)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator>
  _OutputIterator **set_intersection** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Output↩
  Iterator __out, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate>
  _OutputIterator **set_intersection** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Output↩
  Iterator __out, _Predicate __pred)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate>
  _OutputIterator **set_intersection** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Output↩
  Iterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  _OIter **set_intersection** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  _OIter **set_intersection** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate>
  _OIter **set_intersection** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate>
  _OIter **set_intersection** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator>
  _OutputIterator **set_symmetric_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2,
  _OutputIterator __out)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator>
  _OutputIterator **set_symmetric_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2,
  _OutputIterator __out, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate>
  _OutputIterator **set_symmetric_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2,
  _OutputIterator __out, _Predicate __pred)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate>
  _OutputIterator **set_symmetric_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2,
  _OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  _OIter **set_symmetric_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  _OIter **set_symmetric_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate>
  _OIter **set_symmetric_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate>
  _OIter **set_symmetric_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator>
  _OutputIterator **set_union** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator
  __out)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator>
  _OutputIterator **set_union** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator
  __out, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate>
  _OutputIterator **set_union** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator
  __out, _Predicate __pred)

- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate>
  _OutputIterator **set_union** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  _OIter **set_union** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  _OIter **set_union** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate>
  _OIter **set_union** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate>
  _OIter **set_union** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)
- template<typename _RAIter>
  void **sort** (_RAIter __begin, _RAIter __end)
- template<typename _RAIter>
  void **sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::balanced_quicksort_tag __parallelism)
- template<typename _RAIter>
  void **sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::default_parallel_tag __parallelism)
- template<typename _RAIter>
  void **sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_exact_tag __parallelism)
- template<typename _RAIter>
  void **sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_sampling_tag __parallelism)
- template<typename _RAIter>
  void **sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_tag __parallelism)
- template<typename _RAIter>
  void **sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::parallel_tag __parallelism)
- template<typename _RAIter>
  void **sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::quicksort_tag __parallelism)
- template<typename _RAIter>
  void **sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter, typename _Compare>
  void **sort** (_RAIter __begin, _RAIter __end, _Compare __comp)
- template<typename _RAIter, typename _Compare>
  void **sort** (_RAIter __begin, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIter, typename _Compare, typename _Parallelism>
  void **sort** (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)
- template<typename _RAIter>
  void **stable_sort** (_RAIter __begin, _RAIter __end)
- template<typename _RAIter>
  void **stable_sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::balanced_quicksort_tag __parallelism)
- template<typename _RAIter>
  void **stable_sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::default_parallel_tag __parallelism)
- template<typename _RAIter>
  void **stable_sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_tag __parallelism)
- template<typename _RAIter>
  void **stable_sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::parallel_tag __parallelism)
- template<typename _RAIter>
  void **stable_sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::quicksort_tag __parallelism)
- template<typename _RAIter>
  void **stable_sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter, typename _Compare>
  void **stable_sort** (_RAIter __begin, _RAIter __end, _Compare __comp)
- template<typename _RAIter, typename _Compare>
  void **stable_sort** (_RAIter __begin, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)

- template<typename _RAIter, typename _Compare, typename _Parallelism>
  void **stable_sort** (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)
- template<typename _IIter, typename _OutputIterator, typename _UnaryOperation>
  _OutputIterator **transform** (_IIter __begin, _IIter __end, _OutputIterator __result, _UnaryOperation __unary_op)
- template<typename _IIter, typename _OutputIterator, typename _UnaryOperation>
  _OutputIterator **transform** (_IIter __begin, _IIter __end, _OutputIterator __result, _UnaryOperation __unary_op, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter, typename _OutputIterator, typename _UnaryOperation>
  _OutputIterator **transform** (_IIter __begin, _IIter __end, _OutputIterator __result, _UnaryOperation __unary_op, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OIter, typename _UnaryOperation>
  _OIter **transform** (_IIter, _IIter, _OIter, _UnaryOperation)
- template<typename _IIter, typename _OIter, typename _UnaryOperation>
  _OIter **transform** (_IIter, _IIter, _OIter, _UnaryOperation, __gnu_parallel::_Parallelism)
- template<typename _IIter, typename _OIter, typename _UnaryOperation>
  _OIter **transform** (_IIter, _IIter, _OIter, _UnaryOperation, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator **transform** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator **transform** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator **transform** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _BiOperation>
  _OIter **transform** (_IIter1, _IIter1, _IIter2, _OIter, _BiOperation)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _BiOperation>
  _OIter **transform** (_IIter1, _IIter1, _IIter2, _OIter, _BiOperation, __gnu_parallel::_Parallelism)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _BiOperation>
  _OIter **transform** (_IIter1, _IIter1, _IIter2, _OIter, _BiOperation, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OutputIterator>
  _OutputIterator **unique_copy** (_IIter __begin1, _IIter __end1, _OutputIterator __out)
- template<typename _IIter, typename _OutputIterator>
  _OutputIterator **unique_copy** (_IIter __begin1, _IIter __end1, _OutputIterator __out, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OutputIterator, typename _Predicate>
  _OutputIterator **unique_copy** (_IIter __begin1, _IIter __end1, _OutputIterator __out, _Predicate __pred)
- template<typename _IIter, typename _OutputIterator, typename _Predicate>
  _OutputIterator **unique_copy** (_IIter __begin1, _IIter __end1, _OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OIter>
  _OIter **unique_copy** (_IIter, _IIter, _OIter)
- template<typename _IIter, typename _OIter>
  _OIter **unique_copy** (_IIter, _IIter, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OIter, typename _Predicate>
  _OIter **unique_copy** (_IIter, _IIter, _OIter, _Predicate)
- template<typename _IIter, typename _OIter, typename _Predicate>
  _OIter **unique_copy** (_IIter, _IIter, _OIter, _Predicate, __gnu_parallel::sequential_tag)

4.13.1 Detailed Description

GNU parallel code, replaces standard behavior with parallel behavior.

4.13.2 Function Documentation

search()

```
template<typename _ForwardIterator, typename _Searcher>
_ForwardIterator std::__parallel::search (
            _ForwardIterator __first,
            _ForwardIterator __last,
            const _Searcher & __searcher)  [inline]
```

Search a sequence using a Searcher object.

Parameters

__first	A forward iterator.
__last	A forward iterator.
__searcher	A callable object.

Returns

`__searcher(__first,__last)`.first

4.14 std::chrono Namespace Reference

Classes

- class duration
- struct duration_values
- class gps_clock
- class hh_mm_ss
- struct steady_clock
- struct system_clock
- class tai_clock
- class time_point
- struct treat_as_floating_point
- class tzdb_list
- class utc_clock

Typedefs

- using days
- using **file_clock**
- template<typename _Duration>
  using **file_time**
- using **gps_seconds**
- template<typename _Duration>
  using **gps_time**
- using high_resolution_clock
- using hours
- using **local_days**
- using **local_seconds**
- template<typename _Duration>
  using **local_time**
- using microseconds
- using milliseconds

- using minutes
- using months
- using nanoseconds
- using seconds
- using **sys_days**
- using **sys_seconds**
- template<typename _Duration>
  using **sys_time**
- using **tai_seconds**
- template<typename _Duration>
  using **tai_time**
- using **utc_seconds**
- template<typename _Duration>
  using **utc_time**
- using weeks
- using years
- using **zoned_seconds**

Enumerations

- enum class **choose** { **earliest** , **latest** }

Functions

- template<typename _Duration>
  void **__throw_bad_local_time** (const local_time< _Duration > &__tp, const local_info &__i)
- template<typename _Rep, typename _Period>
  constexpr enable_if_t< numeric_limits< _Rep >::is_signed, duration< _Rep, _Period > > abs (duration< _Rep, _Period > __d)
- template<typename _ToDur, typename _Rep, typename _Period>
  constexpr __enable_if_is_duration< _ToDur > ceil (const duration< _Rep, _Period > &__d)
- template<typename _ToDur, typename _Clock, typename _Dur>
  constexpr enable_if_t< __is_duration_v< _ToDur >, time_point< _Clock, _ToDur > > ceil (const time_point< _Clock, _Dur > &__tp)
- template<typename _DestClock, typename _SourceClock, typename _Duration>
  requires __detail::__clock_convs<_DestClock, _SourceClock, _Duration> || __detail::__clock_convs_sys<_DestClock, _SourceClock, ↩ _Duration> || __detail::__clock_convs_utc<_DestClock, _SourceClock, _Duration> || __detail::__clock_convs_sys_utc<_DestClock, _↩ SourceClock, _Duration> || __detail::__clock_convs_utc_sys<_DestClock, _SourceClock, _Duration>
  auto clock_cast (const time_point< _SourceClock, _Duration > &__t)
- const time_zone * **current_zone** ()
- template<typename _ToDur, typename _Rep, typename _Period>
  constexpr __enable_if_is_duration< _ToDur > duration_cast (const duration< _Rep, _Period > &__d)
- template<typename _ToDur, typename _Rep, typename _Period>
  constexpr __enable_if_is_duration< _ToDur > floor (const duration< _Rep, _Period > &__d)
- template<typename _ToDur, typename _Clock, typename _Dur>
  constexpr enable_if_t< __is_duration_v< _ToDur >, time_point< _Clock, _ToDur > > floor (const time_point< _Clock, _Dur > &__tp)
- template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *↩ __fmt, day &__d, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Rep, typename _Period, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *↩ __fmt, duration< _Rep, _Period > &__d, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)

- template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *↩
  __fmt, file_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__↩
  offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *↩
  __fmt, gps_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *_↩
  _offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *↩
  __fmt, local_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *_↩
  _offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *↩
  __fmt, month &__m, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *_↩
  _fmt, month_day &__md, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *↩
  __fmt, sys_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__↩
  offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *↩
  __fmt, tai_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__↩
  offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *↩
  __fmt, utc_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__↩
  offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *↩
  __fmt, weekday &__wd, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *↩
  __fmt, year &__y, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *_↩
  _fmt, year_month &__ym, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, year_month_day &__ymd, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__↩
  offset=nullptr)
- template<typename _Duration>
  leap_second_info **get_leap_second_info** (const utc_time< _Duration > &__ut)
- const tzdb & **get_tzdb** ()
- tzdb_list & **get_tzdb_list** ()
- constexpr bool **is_am** (const hours &__h) noexcept
- constexpr bool **is_pm** (const hours &__h) noexcept
- template<typename _Duration>
  __detail::__local_time_fmt< _Duration > local_time_format (local_time< _Duration > __time, const string *_↩
  _abbrev=nullptr, const seconds *__offset_sec=nullptr)
- const time_zone * **locate_zone** (string_view __tz_name)

- constexpr hours **make12** (const hours &__h) noexcept
- constexpr hours **make24** (const hours &__h, bool __is_pm) noexcept
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const day &__d)
- template<typename _CharT, typename _Traits, typename _Duration>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const file_time< _Duration > &__t)
- template<typename _CharT, typename _Traits, typename _Duration>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const gps_time< _Duration > &__t)
- template<typename _CharT, typename _Traits, typename _Duration>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const hh_mm_ss< _Duration > &__hms)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const local_info &__li)
- template<typename _CharT, typename _Traits, typename _Duration>
  requires requires(const sys_time<_Duration>& __st) { __os << __st; }
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const local_time< _Duration > &__lt)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const month &↩
  __m)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const month_day &__md)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const month_↩
  day_last &__mdl)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const month_↩
  weekday &__mwd)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const month_↩
  weekday_last &__mwdl)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const sys_days &__dp)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const sys_info &__i)
- template<typename _CharT, typename _Traits, typename _Duration>
  requires (!treat_as_floating_point_v<typename _Duration::rep>) && ratio_less_v<typename _Duration::period, days::period>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const sys_time< _Duration > &__tp)
- template<typename _CharT, typename _Traits, typename _Duration>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const tai_time< _Duration > &__t)
- template<typename _CharT, typename _Traits, typename _Duration>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const utc_time< _Duration > &__t)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const weekday &__wd)

- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const weekday_indexed &__wdi)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const weekday_last &__wdl)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const year &__y)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const year_month &__ym)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const year_month_day &__ymd)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const year_month_day_last &__ymdl)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const year_month_weekday &__ymwd)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const year_month_weekday_last &__ymwdl)
- template<typename _CharT, typename _Traits, typename _Duration, typename _TimeZonePtr>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const zoned_time< _Duration, _TimeZonePtr > &__t)
- template<typename _CharT, typename _Traits, typename _Rep, typename _Period>
  basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const duration< _Rep, _Period > &__d)
- template<typename _Dur1, typename _TZPtr1, typename _Dur2, typename _TZPtr2>
  bool **operator==** (const zoned_time< _Dur1, _TZPtr1 > &__x, const zoned_time< _Dur2, _TZPtr2 > &__y)
- template<typename _CharT, __detail::__parsable< _CharT > _Parsable>
  auto **parse** (const _CharT *__fmt, _Parsable &__tp)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits, _Alloc>, __detail::__parsable< _CharT, _Traits, _StrT > _Parsable>
  auto **parse** (const _CharT *__fmt, _Parsable &__tp, basic_string< _CharT, _Traits, _Alloc > &__abbrev)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits, _Alloc>, __detail::__parsable< _CharT, _Traits, _StrT, minutes > _Parsable>
  auto **parse** (const _CharT *__fmt, _Parsable &__tp, basic_string< _CharT, _Traits, _Alloc > &__abbrev, minutes &__offset)
- template<typename _CharT, typename _Traits = char_traits<_CharT>, typename _StrT = basic_string<_CharT, _Traits>, __detail::__parsable< _CharT, _Traits, _StrT, minutes > _Parsable>
  auto **parse** (const _CharT *__fmt, _Parsable &__tp, minutes &__offset)
- template<typename _CharT, typename _Traits, typename _Alloc, __detail::__parsable< _CharT, _Traits > _Parsable>
  auto **parse** (const basic_string< _CharT, _Traits, _Alloc > &__fmt, _Parsable &__tp)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits, _Alloc>, __detail::__parsable< _CharT, _Traits, _StrT > _Parsable>
  auto **parse** (const basic_string< _CharT, _Traits, _Alloc > &__fmt, _Parsable &__tp, basic_string< _CharT, _Traits, _Alloc > &__abbrev)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits, _Alloc>, __detail::__parsable< _CharT, _Traits, _StrT, minutes > _Parsable>
  auto **parse** (const basic_string< _CharT, _Traits, _Alloc > &__fmt, _Parsable &__tp, basic_string< _CharT, _Traits, _Alloc > &__abbrev, minutes &__offset)

- template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits>, __detail::__parsable< _CharT, _Traits, _StrT, minutes > _Parsable>
  auto **parse** (const basic_string< _CharT, _Traits, _Alloc > &__fmt, _Parsable &__tp, minutes &__offset)
- const tzdb & **reload_tzdb** ()
- string **remote_version** ()
- template<typename _ToDur, typename _Rep, typename _Period>
  constexpr enable_if_t< __and_< __is_duration< _ToDur >, __not_< treat_as_floating_point< typename _To↩ Dur::rep > > >::value, _ToDur > round (const duration< _Rep, _Period > &__d)
- template<typename _ToDur, typename _Clock, typename _Dur>
  constexpr enable_if_t< __is_duration_v< _ToDur > &&!treat_as_floating_point_v< typename _ToDur::rep >, time_point< _Clock, _ToDur > > round (const time_point< _Clock, _Dur > &__tp)
- template<typename _ToDur, typename _Clock, typename _Dur>
  constexpr __enable_if_t< __is_duration< _ToDur >::value, time_point< _Clock, _ToDur > > time_point_cast (const time_point< _Clock, _Dur > &__t)
- **zoned_time** () -> zoned_time< seconds >
- template<typename _TimeZonePtrOrName>
  zoned_time (_TimeZonePtrOrName &&) -> zoned_time< seconds, __time_zone_representation< _Time↩ ZonePtrOrName > >
- template<typename _TimeZonePtrOrName, typename _Duration>
  zoned_time (_TimeZonePtrOrName &&, local_time< _Duration >, choose=choose::earliest) -> zoned_time< common_type_t< _Duration, seconds >, __time_zone_representation< _TimeZonePtrOrName > >
- template<typename _TimeZonePtrOrName, typename _Duration>
  zoned_time (_TimeZonePtrOrName &&, sys_time< _Duration >) -> zoned_time< common_type_t< _Duration, seconds >, __time_zone_representation< _TimeZonePtrOrName > >
- template<typename _Duration, typename _TimeZonePtrOrName, typename _TimeZonePtr2>
  zoned_time (_TimeZonePtrOrName &&, zoned_time< _Duration, _TimeZonePtr2 >, choose=choose::earliest) -> zoned_time< common_type_t< _Duration, seconds >, __time_zone_representation< _TimeZonePtrOrName > >
- template<typename _Duration>
  zoned_time (sys_time< _Duration >) -> zoned_time< common_type_t< _Duration, seconds > >

- template<typename _Clock, typename _Dur1, three_way_comparable_with< _Dur1 > _Dur2>
  constexpr auto **operator<=>** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)
- template<typename _Clock, typename _Dur1, typename _Dur2>
  constexpr bool **operator<** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)
- template<typename _Clock, typename _Dur1, typename _Dur2>
  constexpr bool **operator<=** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)
- template<typename _Clock, typename _Dur1, typename _Dur2>
  constexpr bool **operator>** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)
- template<typename _Clock, typename _Dur1, typename _Dur2>
  constexpr bool **operator>=** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)

Variables

- template<typename _Tp, typename _Clock>
  constexpr bool **__is_time_point_for_v**
- template<typename _Clock, typename _Duration>
  constexpr bool **__is_time_point_for_v< time_point< _Clock, _Duration >, _Clock >**

- constexpr month **April**
- constexpr month **August**
- constexpr month **December**
- constexpr month **February**
- constexpr weekday **Friday**
- template<typename _Tp, typename = void>
  constexpr bool **is_clock_v**
- template<typename _Tp>
  constexpr bool **is_clock_v**< **_Tp**, **void_t**< **typename _Tp::rep**, **typename _Tp::period**, **typename _Tp**↩
  ::duration, **typename _Tp::time_point::duration**, **decltype(_Tp::is_steady)**, **decltype(_Tp::now())**> >
- template<> constexpr bool **is_clock_v**< **file_clock** >
- template<> constexpr bool **is_clock_v**< **gps_clock** >
- template<> constexpr bool **is_clock_v**< **steady_clock** >
- template<> constexpr bool **is_clock_v**< **system_clock** >
- template<> constexpr bool **is_clock_v**< **tai_clock** >
- template<> constexpr bool **is_clock_v**< **utc_clock** >
- constexpr month **January**
- constexpr month **July**
- constexpr month **June**
- constexpr last_spec **last**
- constexpr month **March**
- constexpr month **May**
- constexpr weekday **Monday**
- constexpr month **November**
- constexpr month **October**
- constexpr weekday **Saturday**
- constexpr month **September**
- constexpr weekday **Sunday**
- constexpr weekday **Thursday**
- template<typename _Rep>
  constexpr bool **treat_as_floating_point_v**
- template<> constexpr bool **treat_as_floating_point_v**< **double** >
- template<> constexpr bool **treat_as_floating_point_v**< **float** >
- template<> constexpr bool **treat_as_floating_point_v**< **int** >
- template<> constexpr bool **treat_as_floating_point_v**< **long** >
- template<> constexpr bool **treat_as_floating_point_v**< **long double** >
- template<> constexpr bool **treat_as_floating_point_v**< **long long** >
- constexpr weekday **Tuesday**
- constexpr weekday **Wednesday**

- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type operator- (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type operator+ (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)

- template<typename _Rep1, typename _Rep2, typename _Period>
  constexpr duration< __common_rep_t< _Rep2, _Rep1 >, _Period > operator* (const _Rep1 &__s, const duration< _Rep2, _Period > &__d)
- template<typename _Rep1, typename _Period, typename _Rep2>
  constexpr duration< __common_rep_t< _Rep1, __disable_if_is_duration< _Rep2 > >, _Period > operator/ (const duration< _Rep1, _Period > &__d, const _Rep2 &__s)

- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr common_type< _Rep1, _Rep2 >::type operator/ (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Period, typename _Rep2>
  constexpr duration< __common_rep_t< _Rep1, __disable_if_is_duration< _Rep2 > >, _Period > operator% (const duration< _Rep1, _Period > &__d, const _Rep2 &__s)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type operator% (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Period, typename _Rep2>
  constexpr duration< __common_rep_t< _Rep1, _Rep2 >, _Period > operator* (const duration< _Rep1, _↩ Period > &__d, const _Rep2 &__s)

- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr bool operator< (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  requires three_way_comparable<common_type_t<_Rep1, _Rep2>>
  constexpr auto operator<=> (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr bool operator<= (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr bool operator> (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr bool operator>= (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr bool operator== (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)

- template<typename _Rep1, typename _Period1, typename _Clock, typename _Dur2>
  constexpr time_point< _Clock, typename common_type< duration< _Rep1, _Period1 >, _Dur2 >::type > operator+ (const duration< _Rep1, _Period1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)
- template<typename _Clock, typename _Dur1, typename _Rep2, typename _Period2>
  constexpr time_point< _Clock, typename common_type< _Dur1, duration< _Rep2, _Period2 > >::type > operator- (const time_point< _Clock, _Dur1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Clock, typename _Dur1, typename _Dur2>
  constexpr common_type< _Dur1, _Dur2 >::type operator- (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)
- template<typename _Clock, typename _Dur1, typename _Rep2, typename _Period2>
  constexpr time_point< _Clock, typename common_type< _Dur1, duration< _Rep2, _Period2 > >::type > operator+ (const time_point< _Clock, _Dur1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)

4.14.1 Detailed Description

ISO C++ 2011 namespace for date and time utilities.

4.15 std::decimal Namespace Reference

Classes

- class decimal128

- class decimal32
- class decimal64

Functions

- double **decimal128_to_double** (decimal128 __d)
- float **decimal128_to_float** (decimal128 __d)
- long double **decimal128_to_long_double** (decimal128 __d)
- long long **decimal128_to_long_long** (decimal128 __d)
- double **decimal32_to_double** (decimal32 __d)
- float **decimal32_to_float** (decimal32 __d)
- long double **decimal32_to_long_double** (decimal32 __d)
- long long decimal32_to_long_long (decimal32 __d)
- double **decimal64_to_double** (decimal64 __d)
- float **decimal64_to_float** (decimal64 __d)
- long double **decimal64_to_long_double** (decimal64 __d)
- long long **decimal64_to_long_long** (decimal64 __d)
- double **decimal_to_double** (decimal128 __d)
- double **decimal_to_double** (decimal32 __d)
- double **decimal_to_double** (decimal64 __d)
- float **decimal_to_float** (decimal128 __d)
- float **decimal_to_float** (decimal32 __d)
- float **decimal_to_float** (decimal64 __d)
- long double **decimal_to_long_double** (decimal128 __d)
- long double **decimal_to_long_double** (decimal32 __d)
- long double **decimal_to_long_double** (decimal64 __d)
- long long **decimal_to_long_long** (decimal128 __d)
- long long **decimal_to_long_long** (decimal32 __d)
- long long **decimal_to_long_long** (decimal64 __d)
- static decimal128 **make_decimal128** (long long __coeff, int __exp)
- static decimal128 **make_decimal128** (unsigned long long __coeff, int __exp)
- static decimal32 **make_decimal32** (long long __coeff, int __exp)
- static decimal32 **make_decimal32** (unsigned long long __coeff, int __exp)
- static decimal64 **make_decimal64** (long long __coeff, int __exp)
- static decimal64 **make_decimal64** (unsigned long long __coeff, int __exp)
- bool **operator!=** (decimal128 __lhs, decimal128 __rhs)
- bool **operator!=** (decimal128 __lhs, decimal32 __rhs)
- bool **operator!=** (decimal128 __lhs, decimal64 __rhs)
- bool **operator!=** (decimal128 __lhs, int __rhs)
- bool **operator!=** (decimal128 __lhs, long __rhs)
- bool **operator!=** (decimal128 __lhs, long long __rhs)
- bool **operator!=** (decimal128 __lhs, unsigned int __rhs)
- bool **operator!=** (decimal128 __lhs, unsigned long __rhs)
- bool **operator!=** (decimal128 __lhs, unsigned long long __rhs)
- bool **operator!=** (decimal32 __lhs, decimal128 __rhs)
- bool **operator!=** (decimal32 __lhs, decimal32 __rhs)
- bool **operator!=** (decimal32 __lhs, decimal64 __rhs)
- bool **operator!=** (decimal32 __lhs, int __rhs)
- bool **operator!=** (decimal32 __lhs, long __rhs)
- bool **operator!=** (decimal32 __lhs, long long __rhs)
- bool **operator!=** (decimal32 __lhs, unsigned int __rhs)

- bool **operator!=** (decimal32 __lhs, unsigned long __rhs)
- bool **operator!=** (decimal32 __lhs, unsigned long long __rhs)
- bool **operator!=** (decimal64 __lhs, decimal128 __rhs)
- bool **operator!=** (decimal64 __lhs, decimal32 __rhs)
- bool **operator!=** (decimal64 __lhs, decimal64 __rhs)
- bool **operator!=** (decimal64 __lhs, int __rhs)
- bool **operator!=** (decimal64 __lhs, long __rhs)
- bool **operator!=** (decimal64 __lhs, long long __rhs)
- bool **operator!=** (decimal64 __lhs, unsigned int __rhs)
- bool **operator!=** (decimal64 __lhs, unsigned long __rhs)
- bool **operator!=** (decimal64 __lhs, unsigned long long __rhs)
- bool **operator!=** (int __lhs, decimal128 __rhs)
- bool **operator!=** (int __lhs, decimal32 __rhs)
- bool **operator!=** (int __lhs, decimal64 __rhs)
- bool **operator!=** (long __lhs, decimal128 __rhs)
- bool **operator!=** (long __lhs, decimal32 __rhs)
- bool **operator!=** (long __lhs, decimal64 __rhs)
- bool **operator!=** (long long __lhs, decimal128 __rhs)
- bool **operator!=** (long long __lhs, decimal32 __rhs)
- bool **operator!=** (long long __lhs, decimal64 __rhs)
- bool **operator!=** (unsigned int __lhs, decimal128 __rhs)
- bool **operator!=** (unsigned int __lhs, decimal32 __rhs)
- bool **operator!=** (unsigned int __lhs, decimal64 __rhs)
- bool **operator!=** (unsigned long __lhs, decimal128 __rhs)
- bool **operator!=** (unsigned long __lhs, decimal32 __rhs)
- bool **operator!=** (unsigned long __lhs, decimal64 __rhs)
- bool **operator!=** (unsigned long long __lhs, decimal128 __rhs)
- bool **operator!=** (unsigned long long __lhs, decimal32 __rhs)
- bool **operator!=** (unsigned long long __lhs, decimal64 __rhs)
- decimal128 **operator*** (decimal128 __lhs, decimal128 __rhs)
- decimal128 **operator*** (decimal128 __lhs, decimal32 __rhs)
- decimal128 **operator*** (decimal128 __lhs, decimal64 __rhs)
- decimal128 **operator*** (decimal128 __lhs, int __rhs)
- decimal128 **operator*** (decimal128 __lhs, long __rhs)
- decimal128 **operator*** (decimal128 __lhs, long long __rhs)
- decimal128 **operator*** (decimal128 __lhs, unsigned int __rhs)
- decimal128 **operator*** (decimal128 __lhs, unsigned long __rhs)
- decimal128 **operator*** (decimal128 __lhs, unsigned long long __rhs)
- decimal128 **operator*** (decimal32 __lhs, decimal128 __rhs)
- decimal32 **operator*** (decimal32 __lhs, decimal32 __rhs)
- decimal64 **operator*** (decimal32 __lhs, decimal64 __rhs)
- decimal32 **operator*** (decimal32 __lhs, int __rhs)
- decimal32 **operator*** (decimal32 __lhs, long __rhs)
- decimal32 **operator*** (decimal32 __lhs, long long __rhs)
- decimal32 **operator*** (decimal32 __lhs, unsigned int __rhs)
- decimal32 **operator*** (decimal32 __lhs, unsigned long __rhs)
- decimal32 **operator*** (decimal32 __lhs, unsigned long long __rhs)
- decimal128 **operator*** (decimal64 __lhs, decimal128 __rhs)
- decimal64 **operator*** (decimal64 __lhs, decimal32 __rhs)
- decimal64 **operator*** (decimal64 __lhs, decimal64 __rhs)
- decimal64 **operator*** (decimal64 __lhs, int __rhs)

- decimal64 **operator**∗ (decimal64 __lhs, long __rhs)
- decimal64 **operator**∗ (decimal64 __lhs, long long __rhs)
- decimal64 **operator**∗ (decimal64 __lhs, unsigned int __rhs)
- decimal64 **operator**∗ (decimal64 __lhs, unsigned long __rhs)
- decimal64 **operator**∗ (decimal64 __lhs, unsigned long long __rhs)
- decimal128 **operator**∗ (int __lhs, decimal128 __rhs)
- decimal32 **operator**∗ (int __lhs, decimal32 __rhs)
- decimal64 **operator**∗ (int __lhs, decimal64 __rhs)
- decimal128 **operator**∗ (long __lhs, decimal128 __rhs)
- decimal32 **operator**∗ (long __lhs, decimal32 __rhs)
- decimal64 **operator**∗ (long __lhs, decimal64 __rhs)
- decimal128 **operator**∗ (long long __lhs, decimal128 __rhs)
- decimal32 **operator**∗ (long long __lhs, decimal32 __rhs)
- decimal64 **operator**∗ (long long __lhs, decimal64 __rhs)
- decimal128 **operator**∗ (unsigned int __lhs, decimal128 __rhs)
- decimal32 **operator**∗ (unsigned int __lhs, decimal32 __rhs)
- decimal64 **operator**∗ (unsigned int __lhs, decimal64 __rhs)
- decimal128 **operator**∗ (unsigned long __lhs, decimal128 __rhs)
- decimal32 **operator**∗ (unsigned long __lhs, decimal32 __rhs)
- decimal64 **operator**∗ (unsigned long __lhs, decimal64 __rhs)
- decimal128 **operator**∗ (unsigned long long __lhs, decimal128 __rhs)
- decimal32 **operator**∗ (unsigned long long __lhs, decimal32 __rhs)
- decimal64 **operator**∗ (unsigned long long __lhs, decimal64 __rhs)
- decimal128 **operator+** (decimal128 __lhs, decimal128 __rhs)
- decimal128 **operator+** (decimal128 __lhs, decimal32 __rhs)
- decimal128 **operator+** (decimal128 __lhs, decimal64 __rhs)
- decimal128 **operator+** (decimal128 __lhs, int __rhs)
- decimal128 **operator+** (decimal128 __lhs, long __rhs)
- decimal128 **operator+** (decimal128 __lhs, long long __rhs)
- decimal128 **operator+** (decimal128 __lhs, unsigned int __rhs)
- decimal128 **operator+** (decimal128 __lhs, unsigned long __rhs)
- decimal128 **operator+** (decimal128 __lhs, unsigned long long __rhs)
- decimal128 **operator+** (decimal128 __rhs)
- decimal128 **operator+** (decimal32 __lhs, decimal128 __rhs)
- decimal32 **operator+** (decimal32 __lhs, decimal32 __rhs)
- decimal64 **operator+** (decimal32 __lhs, decimal64 __rhs)
- decimal32 **operator+** (decimal32 __lhs, int __rhs)
- decimal32 **operator+** (decimal32 __lhs, long __rhs)
- decimal32 **operator+** (decimal32 __lhs, long long __rhs)
- decimal32 **operator+** (decimal32 __lhs, unsigned int __rhs)
- decimal32 **operator+** (decimal32 __lhs, unsigned long __rhs)
- decimal32 **operator+** (decimal32 __lhs, unsigned long long __rhs)
- decimal32 **operator+** (decimal32 __rhs)
- decimal128 **operator+** (decimal64 __lhs, decimal128 __rhs)
- decimal64 **operator+** (decimal64 __lhs, decimal32 __rhs)
- decimal64 **operator+** (decimal64 __lhs, decimal64 __rhs)
- decimal64 **operator+** (decimal64 __lhs, int __rhs)
- decimal64 **operator+** (decimal64 __lhs, long __rhs)
- decimal64 **operator+** (decimal64 __lhs, long long __rhs)
- decimal64 **operator+** (decimal64 __lhs, unsigned int __rhs)
- decimal64 **operator+** (decimal64 __lhs, unsigned long __rhs)

- decimal64 **operator+** (decimal64 __lhs, unsigned long long __rhs)
- decimal64 **operator+** (decimal64 __rhs)
- decimal128 **operator+** (int __lhs, decimal128 __rhs)
- decimal32 **operator+** (int __lhs, decimal32 __rhs)
- decimal64 **operator+** (int __lhs, decimal64 __rhs)
- decimal128 **operator+** (long __lhs, decimal128 __rhs)
- decimal32 **operator+** (long __lhs, decimal32 __rhs)
- decimal64 **operator+** (long __lhs, decimal64 __rhs)
- decimal128 **operator+** (long long __lhs, decimal128 __rhs)
- decimal32 **operator+** (long long __lhs, decimal32 __rhs)
- decimal64 **operator+** (long long __lhs, decimal64 __rhs)
- decimal128 **operator+** (unsigned int __lhs, decimal128 __rhs)
- decimal32 **operator+** (unsigned int __lhs, decimal32 __rhs)
- decimal64 **operator+** (unsigned int __lhs, decimal64 __rhs)
- decimal128 **operator+** (unsigned long __lhs, decimal128 __rhs)
- decimal32 **operator+** (unsigned long __lhs, decimal32 __rhs)
- decimal64 **operator+** (unsigned long __lhs, decimal64 __rhs)
- decimal128 **operator+** (unsigned long long __lhs, decimal128 __rhs)
- decimal32 **operator+** (unsigned long long __lhs, decimal32 __rhs)
- decimal64 **operator+** (unsigned long long __lhs, decimal64 __rhs)
- decimal128 **operator-** (decimal128 __lhs, decimal128 __rhs)
- decimal128 **operator-** (decimal128 __lhs, decimal32 __rhs)
- decimal128 **operator-** (decimal128 __lhs, decimal64 __rhs)
- decimal128 **operator-** (decimal128 __lhs, int __rhs)
- decimal128 **operator-** (decimal128 __lhs, long __rhs)
- decimal128 **operator-** (decimal128 __lhs, long long __rhs)
- decimal128 **operator-** (decimal128 __lhs, unsigned int __rhs)
- decimal128 **operator-** (decimal128 __lhs, unsigned long __rhs)
- decimal128 **operator-** (decimal128 __lhs, unsigned long long __rhs)
- decimal128 **operator-** (decimal128 __rhs)
- decimal128 **operator-** (decimal32 __lhs, decimal128 __rhs)
- decimal32 **operator-** (decimal32 __lhs, decimal32 __rhs)
- decimal64 **operator-** (decimal32 __lhs, decimal64 __rhs)
- decimal32 **operator-** (decimal32 __lhs, int __rhs)
- decimal32 **operator-** (decimal32 __lhs, long __rhs)
- decimal32 **operator-** (decimal32 __lhs, long long __rhs)
- decimal32 **operator-** (decimal32 __lhs, unsigned int __rhs)
- decimal32 **operator-** (decimal32 __lhs, unsigned long __rhs)
- decimal32 **operator-** (decimal32 __lhs, unsigned long long __rhs)
- decimal32 **operator-** (decimal32 __rhs)
- decimal128 **operator-** (decimal64 __lhs, decimal128 __rhs)
- decimal64 **operator-** (decimal64 __lhs, decimal32 __rhs)
- decimal64 **operator-** (decimal64 __lhs, decimal64 __rhs)
- decimal64 **operator-** (decimal64 __lhs, int __rhs)
- decimal64 **operator-** (decimal64 __lhs, long __rhs)
- decimal64 **operator-** (decimal64 __lhs, long long __rhs)
- decimal64 **operator-** (decimal64 __lhs, unsigned int __rhs)
- decimal64 **operator-** (decimal64 __lhs, unsigned long __rhs)
- decimal64 **operator-** (decimal64 __lhs, unsigned long long __rhs)
- decimal64 **operator-** (decimal64 __rhs)
- decimal128 **operator-** (int __lhs, decimal128 __rhs)

- decimal32 **operator-** (int __lhs, decimal32 __rhs)
- decimal64 **operator-** (int __lhs, decimal64 __rhs)
- decimal128 **operator-** (long __lhs, decimal128 __rhs)
- decimal32 **operator-** (long __lhs, decimal32 __rhs)
- decimal64 **operator-** (long __lhs, decimal64 __rhs)
- decimal128 **operator-** (long long __lhs, decimal128 __rhs)
- decimal32 **operator-** (long long __lhs, decimal32 __rhs)
- decimal64 **operator-** (long long __lhs, decimal64 __rhs)
- decimal128 **operator-** (unsigned int __lhs, decimal128 __rhs)
- decimal32 **operator-** (unsigned int __lhs, decimal32 __rhs)
- decimal64 **operator-** (unsigned int __lhs, decimal64 __rhs)
- decimal128 **operator-** (unsigned long __lhs, decimal128 __rhs)
- decimal32 **operator-** (unsigned long __lhs, decimal32 __rhs)
- decimal64 **operator-** (unsigned long __lhs, decimal64 __rhs)
- decimal128 **operator-** (unsigned long long __lhs, decimal128 __rhs)
- decimal32 **operator-** (unsigned long long __lhs, decimal32 __rhs)
- decimal64 **operator-** (unsigned long long __lhs, decimal64 __rhs)
- decimal128 **operator/** (decimal128 __lhs, decimal128 __rhs)
- decimal128 **operator/** (decimal128 __lhs, decimal32 __rhs)
- decimal128 **operator/** (decimal128 __lhs, decimal64 __rhs)
- decimal128 **operator/** (decimal128 __lhs, int __rhs)
- decimal128 **operator/** (decimal128 __lhs, long __rhs)
- decimal128 **operator/** (decimal128 __lhs, long long __rhs)
- decimal128 **operator/** (decimal128 __lhs, unsigned int __rhs)
- decimal128 **operator/** (decimal128 __lhs, unsigned long __rhs)
- decimal128 **operator/** (decimal128 __lhs, unsigned long long __rhs)
- decimal128 **operator/** (decimal32 __lhs, decimal128 __rhs)
- decimal32 **operator/** (decimal32 __lhs, decimal32 __rhs)
- decimal64 **operator/** (decimal32 __lhs, decimal64 __rhs)
- decimal32 **operator/** (decimal32 __lhs, int __rhs)
- decimal32 **operator/** (decimal32 __lhs, long __rhs)
- decimal32 **operator/** (decimal32 __lhs, long long __rhs)
- decimal32 **operator/** (decimal32 __lhs, unsigned int __rhs)
- decimal32 **operator/** (decimal32 __lhs, unsigned long __rhs)
- decimal32 **operator/** (decimal32 __lhs, unsigned long long __rhs)
- decimal128 **operator/** (decimal64 __lhs, decimal128 __rhs)
- decimal64 **operator/** (decimal64 __lhs, decimal32 __rhs)
- decimal64 **operator/** (decimal64 __lhs, decimal64 __rhs)
- decimal64 **operator/** (decimal64 __lhs, int __rhs)
- decimal64 **operator/** (decimal64 __lhs, long __rhs)
- decimal64 **operator/** (decimal64 __lhs, long long __rhs)
- decimal64 **operator/** (decimal64 __lhs, unsigned int __rhs)
- decimal64 **operator/** (decimal64 __lhs, unsigned long __rhs)
- decimal64 **operator/** (decimal64 __lhs, unsigned long long __rhs)
- decimal128 **operator/** (int __lhs, decimal128 __rhs)
- decimal32 **operator/** (int __lhs, decimal32 __rhs)
- decimal64 **operator/** (int __lhs, decimal64 __rhs)
- decimal128 **operator/** (long __lhs, decimal128 __rhs)
- decimal32 **operator/** (long __lhs, decimal32 __rhs)
- decimal64 **operator/** (long __lhs, decimal64 __rhs)
- decimal128 **operator/** (long long __lhs, decimal128 __rhs)

- decimal32 **operator**/ (long long __lhs, decimal32 __rhs)
- decimal64 **operator**/ (long long __lhs, decimal64 __rhs)
- decimal128 **operator**/ (unsigned int __lhs, decimal128 __rhs)
- decimal32 **operator**/ (unsigned int __lhs, decimal32 __rhs)
- decimal64 **operator**/ (unsigned int __lhs, decimal64 __rhs)
- decimal128 **operator**/ (unsigned long __lhs, decimal128 __rhs)
- decimal32 **operator**/ (unsigned long __lhs, decimal32 __rhs)
- decimal64 **operator**/ (unsigned long __lhs, decimal64 __rhs)
- decimal128 **operator**/ (unsigned long long __lhs, decimal128 __rhs)
- decimal32 **operator**/ (unsigned long long __lhs, decimal32 __rhs)
- decimal64 **operator**/ (unsigned long long __lhs, decimal64 __rhs)
- bool **operator**< (decimal128 __lhs, decimal128 __rhs)
- bool **operator**< (decimal128 __lhs, decimal32 __rhs)
- bool **operator**< (decimal128 __lhs, decimal64 __rhs)
- bool **operator**< (decimal128 __lhs, int __rhs)
- bool **operator**< (decimal128 __lhs, long __rhs)
- bool **operator**< (decimal128 __lhs, long long __rhs)
- bool **operator**< (decimal128 __lhs, unsigned int __rhs)
- bool **operator**< (decimal128 __lhs, unsigned long __rhs)
- bool **operator**< (decimal128 __lhs, unsigned long long __rhs)
- bool **operator**< (decimal32 __lhs, decimal128 __rhs)
- bool **operator**< (decimal32 __lhs, decimal32 __rhs)
- bool **operator**< (decimal32 __lhs, decimal64 __rhs)
- bool **operator**< (decimal32 __lhs, int __rhs)
- bool **operator**< (decimal32 __lhs, long __rhs)
- bool **operator**< (decimal32 __lhs, long long __rhs)
- bool **operator**< (decimal32 __lhs, unsigned int __rhs)
- bool **operator**< (decimal32 __lhs, unsigned long __rhs)
- bool **operator**< (decimal32 __lhs, unsigned long long __rhs)
- bool **operator**< (decimal64 __lhs, decimal128 __rhs)
- bool **operator**< (decimal64 __lhs, decimal32 __rhs)
- bool **operator**< (decimal64 __lhs, decimal64 __rhs)
- bool **operator**< (decimal64 __lhs, int __rhs)
- bool **operator**< (decimal64 __lhs, long __rhs)
- bool **operator**< (decimal64 __lhs, long long __rhs)
- bool **operator**< (decimal64 __lhs, unsigned int __rhs)
- bool **operator**< (decimal64 __lhs, unsigned long __rhs)
- bool **operator**< (decimal64 __lhs, unsigned long long __rhs)
- bool **operator**< (int __lhs, decimal128 __rhs)
- bool **operator**< (int __lhs, decimal32 __rhs)
- bool **operator**< (int __lhs, decimal64 __rhs)
- bool **operator**< (long __lhs, decimal128 __rhs)
- bool **operator**< (long __lhs, decimal32 __rhs)
- bool **operator**< (long __lhs, decimal64 __rhs)
- bool **operator**< (long long __lhs, decimal128 __rhs)
- bool **operator**< (long long __lhs, decimal32 __rhs)
- bool **operator**< (long long __lhs, decimal64 __rhs)
- bool **operator**< (unsigned int __lhs, decimal128 __rhs)
- bool **operator**< (unsigned int __lhs, decimal32 __rhs)
- bool **operator**< (unsigned int __lhs, decimal64 __rhs)
- bool **operator**< (unsigned long __lhs, decimal128 __rhs)

- bool **operator**< (unsigned long __lhs, decimal32 __rhs)
- bool **operator**< (unsigned long __lhs, decimal64 __rhs)
- bool **operator**< (unsigned long long __lhs, decimal128 __rhs)
- bool **operator**< (unsigned long long __lhs, decimal32 __rhs)
- bool **operator**< (unsigned long long __lhs, decimal64 __rhs)
- bool **operator==** (decimal128 __lhs, decimal128 __rhs)
- bool **operator==** (decimal128 __lhs, decimal32 __rhs)
- bool **operator==** (decimal128 __lhs, decimal64 __rhs)
- bool **operator==** (decimal128 __lhs, int __rhs)
- bool **operator==** (decimal128 __lhs, long __rhs)
- bool **operator==** (decimal128 __lhs, long long __rhs)
- bool **operator==** (decimal128 __lhs, unsigned int __rhs)
- bool **operator==** (decimal128 __lhs, unsigned long __rhs)
- bool **operator==** (decimal128 __lhs, unsigned long long __rhs)
- bool **operator==** (decimal32 __lhs, decimal128 __rhs)
- bool **operator==** (decimal32 __lhs, decimal32 __rhs)
- bool **operator==** (decimal32 __lhs, decimal64 __rhs)
- bool **operator==** (decimal32 __lhs, int __rhs)
- bool **operator==** (decimal32 __lhs, long __rhs)
- bool **operator==** (decimal32 __lhs, long long __rhs)
- bool **operator==** (decimal32 __lhs, unsigned int __rhs)
- bool **operator==** (decimal32 __lhs, unsigned long __rhs)
- bool **operator==** (decimal32 __lhs, unsigned long long __rhs)
- bool **operator==** (decimal64 __lhs, decimal128 __rhs)
- bool **operator==** (decimal64 __lhs, decimal32 __rhs)
- bool **operator==** (decimal64 __lhs, decimal64 __rhs)
- bool **operator==** (decimal64 __lhs, int __rhs)
- bool **operator==** (decimal64 __lhs, long __rhs)
- bool **operator==** (decimal64 __lhs, long long __rhs)
- bool **operator==** (decimal64 __lhs, unsigned int __rhs)
- bool **operator==** (decimal64 __lhs, unsigned long __rhs)
- bool **operator==** (decimal64 __lhs, unsigned long long __rhs)
- bool **operator==** (int __lhs, decimal128 __rhs)
- bool **operator==** (int __lhs, decimal32 __rhs)
- bool **operator==** (int __lhs, decimal64 __rhs)
- bool **operator==** (long __lhs, decimal128 __rhs)
- bool **operator==** (long __lhs, decimal32 __rhs)
- bool **operator==** (long __lhs, decimal64 __rhs)
- bool **operator==** (long long __lhs, decimal128 __rhs)
- bool **operator==** (long long __lhs, decimal32 __rhs)
- bool **operator==** (long long __lhs, decimal64 __rhs)
- bool **operator==** (unsigned int __lhs, decimal128 __rhs)
- bool **operator==** (unsigned int __lhs, decimal32 __rhs)
- bool **operator==** (unsigned int __lhs, decimal64 __rhs)
- bool **operator==** (unsigned long __lhs, decimal128 __rhs)
- bool **operator==** (unsigned long __lhs, decimal32 __rhs)
- bool **operator==** (unsigned long __lhs, decimal64 __rhs)
- bool **operator==** (unsigned long long __lhs, decimal128 __rhs)
- bool **operator==** (unsigned long long __lhs, decimal32 __rhs)
- bool **operator==** (unsigned long long __lhs, decimal64 __rhs)
- bool **operator**> (decimal128 __lhs, decimal128 __rhs)

- bool **operator**> (decimal128 __lhs, decimal32 __rhs)
- bool **operator**> (decimal128 __lhs, decimal64 __rhs)
- bool **operator**> (decimal128 __lhs, int __rhs)
- bool **operator**> (decimal128 __lhs, long __rhs)
- bool **operator**> (decimal128 __lhs, long long __rhs)
- bool **operator**> (decimal128 __lhs, unsigned int __rhs)
- bool **operator**> (decimal128 __lhs, unsigned long __rhs)
- bool **operator**> (decimal128 __lhs, unsigned long long __rhs)
- bool **operator**> (decimal32 __lhs, decimal128 __rhs)
- bool **operator**> (decimal32 __lhs, decimal32 __rhs)
- bool **operator**> (decimal32 __lhs, decimal64 __rhs)
- bool **operator**> (decimal32 __lhs, int __rhs)
- bool **operator**> (decimal32 __lhs, long __rhs)
- bool **operator**> (decimal32 __lhs, long long __rhs)
- bool **operator**> (decimal32 __lhs, unsigned int __rhs)
- bool **operator**> (decimal32 __lhs, unsigned long __rhs)
- bool **operator**> (decimal32 __lhs, unsigned long long __rhs)
- bool **operator**> (decimal64 __lhs, decimal128 __rhs)
- bool **operator**> (decimal64 __lhs, decimal32 __rhs)
- bool **operator**> (decimal64 __lhs, decimal64 __rhs)
- bool **operator**> (decimal64 __lhs, int __rhs)
- bool **operator**> (decimal64 __lhs, long __rhs)
- bool **operator**> (decimal64 __lhs, long long __rhs)
- bool **operator**> (decimal64 __lhs, unsigned int __rhs)
- bool **operator**> (decimal64 __lhs, unsigned long __rhs)
- bool **operator**> (decimal64 __lhs, unsigned long long __rhs)
- bool **operator**> (int __lhs, decimal128 __rhs)
- bool **operator**> (int __lhs, decimal32 __rhs)
- bool **operator**> (int __lhs, decimal64 __rhs)
- bool **operator**> (long __lhs, decimal128 __rhs)
- bool **operator**> (long __lhs, decimal32 __rhs)
- bool **operator**> (long __lhs, decimal64 __rhs)
- bool **operator**> (long long __lhs, decimal128 __rhs)
- bool **operator**> (long long __lhs, decimal32 __rhs)
- bool **operator**> (long long __lhs, decimal64 __rhs)
- bool **operator**> (unsigned int __lhs, decimal128 __rhs)
- bool **operator**> (unsigned int __lhs, decimal32 __rhs)
- bool **operator**> (unsigned int __lhs, decimal64 __rhs)
- bool **operator**> (unsigned long __lhs, decimal128 __rhs)
- bool **operator**> (unsigned long __lhs, decimal32 __rhs)
- bool **operator**> (unsigned long __lhs, decimal64 __rhs)
- bool **operator**> (unsigned long long __lhs, decimal128 __rhs)
- bool **operator**> (unsigned long long __lhs, decimal32 __rhs)
- bool **operator**> (unsigned long long __lhs, decimal64 __rhs)
- bool **operator**>**=** (decimal128 __lhs, decimal128 __rhs)
- bool **operator**>**=** (decimal128 __lhs, decimal32 __rhs)
- bool **operator**>**=** (decimal128 __lhs, decimal64 __rhs)
- bool **operator**>**=** (decimal128 __lhs, int __rhs)
- bool **operator**>**=** (decimal128 __lhs, long __rhs)
- bool **operator**>**=** (decimal128 __lhs, long long __rhs)
- bool **operator**>**=** (decimal128 __lhs, unsigned int __rhs)

- bool **operator>=** (decimal128 __lhs, unsigned long __rhs)
- bool **operator>=** (decimal128 __lhs, unsigned long long __rhs)
- bool **operator>=** (decimal32 __lhs, decimal128 __rhs)
- bool **operator>=** (decimal32 __lhs, decimal32 __rhs)
- bool **operator>=** (decimal32 __lhs, decimal64 __rhs)
- bool **operator>=** (decimal32 __lhs, int __rhs)
- bool **operator>=** (decimal32 __lhs, long __rhs)
- bool **operator>=** (decimal32 __lhs, long long __rhs)
- bool **operator>=** (decimal32 __lhs, unsigned int __rhs)
- bool **operator>=** (decimal32 __lhs, unsigned long __rhs)
- bool **operator>=** (decimal32 __lhs, unsigned long long __rhs)
- bool **operator>=** (decimal64 __lhs, decimal128 __rhs)
- bool **operator>=** (decimal64 __lhs, decimal32 __rhs)
- bool **operator>=** (decimal64 __lhs, decimal64 __rhs)
- bool **operator>=** (decimal64 __lhs, int __rhs)
- bool **operator>=** (decimal64 __lhs, long __rhs)
- bool **operator>=** (decimal64 __lhs, long long __rhs)
- bool **operator>=** (decimal64 __lhs, unsigned int __rhs)
- bool **operator>=** (decimal64 __lhs, unsigned long __rhs)
- bool **operator>=** (decimal64 __lhs, unsigned long long __rhs)
- bool **operator>=** (int __lhs, decimal128 __rhs)
- bool **operator>=** (int __lhs, decimal32 __rhs)
- bool **operator>=** (int __lhs, decimal64 __rhs)
- bool **operator>=** (long __lhs, decimal128 __rhs)
- bool **operator>=** (long __lhs, decimal32 __rhs)
- bool **operator>=** (long __lhs, decimal64 __rhs)
- bool **operator>=** (long long __lhs, decimal128 __rhs)
- bool **operator>=** (long long __lhs, decimal32 __rhs)
- bool **operator>=** (long long __lhs, decimal64 __rhs)
- bool **operator>=** (unsigned int __lhs, decimal128 __rhs)
- bool **operator>=** (unsigned int __lhs, decimal32 __rhs)
- bool **operator>=** (unsigned int __lhs, decimal64 __rhs)
- bool **operator>=** (unsigned long __lhs, decimal128 __rhs)
- bool **operator>=** (unsigned long __lhs, decimal32 __rhs)
- bool **operator>=** (unsigned long __lhs, decimal64 __rhs)
- bool **operator>=** (unsigned long long __lhs, decimal128 __rhs)
- bool **operator>=** (unsigned long long __lhs, decimal32 __rhs)
- bool **operator>=** (unsigned long long __lhs, decimal64 __rhs)

4.15.1 Detailed Description

ISO/IEC TR 24733 Decimal floating-point arithmetic.

4.15.2 Function Documentation

decimal32_to_long_long()

```
long long std::decimal::decimal32_to_long_long (
            decimal32 __d)
```

Non-conforming extension: Conversion to integral type.

4.16 std::experimental Namespace Reference

Classes

- class any
- class bad_any_cast
- class bad_optional_access
- class basic_string_view
- struct in_place_t
- struct nullopt_t
- class optional
- class ostream_joiner
- struct owner_less< shared_ptr< _Tp > >
- struct owner_less< weak_ptr< _Tp > >
- class propagate_const

Typedefs

- template<typename _RAIter, typename _Hash, typename _Pred, typename _Val = typename iterator_traits<_RAIter>::value_type, typename _Diff = typename iterator_traits<_RAIter>::difference_type>
  using **__boyer_moore_base_t**
- template<typename _Tp>
  using **__propagate_const_elem_type**
- using **erased_type**
- using **string_view**
- using **u16string_view**
- using **u32string_view**
- using **wstring_view**

Functions

- template<typename _Fn, typename _Tuple, std::size_t... _Idx>
  constexpr decltype(auto) **__apply_impl** (_Fn &&__f, _Tuple &&__t, std::index_sequence< _Idx... >)
- template<typename _Tp, size_t _Nm, size_t... _Idx>
  constexpr array< remove_cv_t< _Tp >, _Nm > **__to_array** (_Tp(&__a)[_Nm], index_sequence< _Idx... >)
- std::default_random_engine & **_S_randint_engine** ()
- template<typename _ValueType>
  _ValueType any_cast (const any &__any)
- template<typename _Fn, typename _Tuple>
  constexpr decltype(auto) **apply** (_Fn &&__f, _Tuple &&__t)
- template<typename _Tp>
  bool **atomic_compare_exchange_strong** (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w)
- template<typename _Tp>
  bool **atomic_compare_exchange_strong_explicit** (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w, memory_order __success, memory_order __failure)
- template<typename _Tp>
  bool **atomic_compare_exchange_weak** (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w)
- template<typename _Tp>
  bool **atomic_compare_exchange_weak_explicit** (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w, memory_order __success, memory_order __failure)
- template<typename _Tp>
  void **atomic_exchange** (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r)

- template<typename _Tp>
  shared_ptr< _Tp > **atomic_exchange_explicit** (const shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r, memory_order __mo)
- template<typename _Tp>
  bool **atomic_is_lock_free** (const shared_ptr< _Tp > *__p)
- template<typename _Tp>
  shared_ptr< _Tp > **atomic_load** (const shared_ptr< _Tp > *__p)
- template<typename _Tp>
  shared_ptr< _Tp > **atomic_load_explicit** (const shared_ptr< _Tp > *__p, memory_order __mo)
- template<typename _Tp>
  void **atomic_store** (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r)
- template<typename _Tp>
  shared_ptr< _Tp > **atomic_store_explicit** (const shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r, memory_order __mo)
- template<typename _Tp, typename _Tp1>
  shared_ptr< _Tp > **const_pointer_cast** (const shared_ptr< _Tp1 > &__r) noexcept
- template<typename _Tp, typename _Tp1>
  shared_ptr< _Tp > **dynamic_pointer_cast** (const shared_ptr< _Tp1 > &__r) noexcept
- template<typename _CharT, typename _Traits, typename _Alloc, typename _Up>
  void **erase** (basic_string< _CharT, _Traits, _Alloc > &__cont, const _Up &__value)
- template<typename _Tp, typename _Alloc, typename _Up>
  void **erase** (deque< _Tp, _Alloc > &__cont, const _Up &__value)
- template<typename _Tp, typename _Alloc, typename _Up>
  void **erase** (forward_list< _Tp, _Alloc > &__cont, const _Up &__value)
- template<typename _Tp, typename _Alloc, typename _Up>
  void **erase** (list< _Tp, _Alloc > &__cont, const _Up &__value)
- template<typename _Tp, typename _Alloc, typename _Up>
  void **erase** (vector< _Tp, _Alloc > &__cont, const _Up &__value)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _Predicate>
  void **erase_if** (basic_string< _CharT, _Traits, _Alloc > &__cont, _Predicate __pred)
- template<typename _Tp, typename _Alloc, typename _Predicate>
  void **erase_if** (deque< _Tp, _Alloc > &__cont, _Predicate __pred)
- template<typename _Tp, typename _Alloc, typename _Predicate>
  void **erase_if** (forward_list< _Tp, _Alloc > &__cont, _Predicate __pred)
- template<typename _Tp, typename _Alloc, typename _Predicate>
  void **erase_if** (list< _Tp, _Alloc > &__cont, _Predicate __pred)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc, typename _Predicate>
  void **erase_if** (map< _Key, _Tp, _Compare, _Alloc > &__cont, _Predicate __pred)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc, typename _Predicate>
  void **erase_if** (multimap< _Key, _Tp, _Compare, _Alloc > &__cont, _Predicate __pred)
- template<typename _Key, typename _Compare, typename _Alloc, typename _Predicate>
  void **erase_if** (multiset< _Key, _Compare, _Alloc > &__cont, _Predicate __pred)
- template<typename _Key, typename _Compare, typename _Alloc, typename _Predicate>
  void **erase_if** (set< _Key, _Compare, _Alloc > &__cont, _Predicate __pred)
- template<typename _Key, typename _Tp, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>
  void **erase_if** (unordered_map< _Key, _Tp, _Hash, _CPred, _Alloc > &__cont, _Predicate __pred)
- template<typename _Key, typename _Tp, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>
  void **erase_if** (unordered_multimap< _Key, _Tp, _Hash, _CPred, _Alloc > &__cont, _Predicate __pred)
- template<typename _Key, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>
  void **erase_if** (unordered_multiset< _Key, _Hash, _CPred, _Alloc > &__cont, _Predicate __pred)
- template<typename _Key, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>
  void **erase_if** (unordered_set< _Key, _Hash, _CPred, _Alloc > &__cont, _Predicate __pred)

- template<typename _Tp, typename _Alloc, typename _Predicate>
  void **erase_if** (vector< _Tp, _Alloc > &__cont, _Predicate __pred)
- template<typename _Mn, typename _Nn>
  constexpr common_type_t< _Mn, _Nn > gcd (_Mn __m, _Nn __n) noexcept
- template<typename _Del, typename _Tp>
  _Del * get_deleter (const shared_ptr< _Tp > &__p) noexcept
- template<typename _Tp>
  constexpr const _Tp & **get_underlying** (const propagate_const< _Tp > &__pt) noexcept
- template<typename _Tp>
  constexpr _Tp & **get_underlying** (propagate_const< _Tp > &__pt) noexcept
- template<typename _Mn, typename _Nn>
  constexpr common_type_t< _Mn, _Nn > lcm (_Mn __m, _Nn __n)
- template<typename _Dest = void, typename... _Types>
  constexpr array< typename __make_array_elem< _Dest, _Types... >::type, sizeof...(_Types)> make_array (↩
  _Types &&... __t)
- template<typename _RAIter, typename _Hash = std::hash<typename std::iterator_traits<_RAIter>::value_type>, typename _Binary↩
  Predicate = equal_to<>>
  boyer_moore_horspool_searcher< _RAIter, _Hash, _BinaryPredicate > make_boyer_moore_horspool_searcher
  (_RAIter __pat_first, _RAIter __pat_last, _Hash __hf=_Hash(), _BinaryPredicate __pred=_BinaryPredicate())
- template<typename _RAIter, typename _Hash = std::hash<typename std::iterator_traits<_RAIter>::value_type>, typename _Binary↩
  Predicate = equal_to<>>
  boyer_moore_searcher< _RAIter, _Hash, _BinaryPredicate > make_boyer_moore_searcher (_RAIter __pat_↩
  first, _RAIter __pat_last, _Hash __hf=_Hash(), _BinaryPredicate __pred=_BinaryPredicate())
- template<typename _ForwardIterator, typename _BinaryPredicate = std::equal_to<>>
  default_searcher< _ForwardIterator, _BinaryPredicate > make_default_searcher (_ForwardIterator __pat_first,
  _ForwardIterator __pat_last, _BinaryPredicate __pred=_BinaryPredicate())
- template<typename _Tp>
  constexpr observer_ptr< _Tp > **make_observer** (_Tp *__p) noexcept
- template<typename _CharT, typename _Traits, typename _DelimT>
  ostream_joiner< decay_t< _DelimT >, _CharT, _Traits > make_ostream_joiner (basic_ostream< _CharT, _↩
  Traits > &__os, _DelimT &&__delimiter)
- template<typename _Fn>
  auto not_fn (_Fn &&__fn) noexcept(std::is_nothrow_constructible< std::decay_t< _Fn >, _Fn && >::value)
- template<typename _CharT, typename _Traits>
  constexpr bool **operator!=** (__type_identity_t< basic_string_view< _CharT, _Traits > > __x, basic_string_view<
  _CharT, _Traits > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **operator!=** (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view<
  _CharT, _Traits > > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **operator!=** (basic_string_view< _CharT, _Traits > __x, basic_string_view< _CharT, _Traits >
  __y) noexcept
- template<typename _Tp, typename _Up>
  constexpr bool **operator!=** (const _Tp &__t, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **operator!=** (const propagate_const< _Tp > &__pt, const _Up &__u)
- template<typename _Tp, typename _Up>
  constexpr bool **operator!=** (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)
- template<typename _Tp>
  constexpr bool **operator!=** (const propagate_const< _Tp > &__pt, nullptr_t)
- template<typename _Tp>
  bool **operator!=** (const shared_ptr< _Tp > &__a, nullptr_t) noexcept

- template<typename _Tp1, typename _Tp2>
  bool **operator!=** (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept
- template<typename _Tp>
  constexpr bool **operator!=** (nullptr_t, const propagate_const< _Tp > &__pu)
- template<typename _Tp>
  bool **operator!=** (nullptr_t, const shared_ptr< _Tp > &__a) noexcept
- template<typename _Tp>
  constexpr bool **operator!=** (nullptr_t, observer_ptr< _Tp > __p) noexcept
- template<typename _Tp>
  bool **operator!=** (observer_ptr< _Tp > __p, nullptr_t) noexcept
- template<typename _Tp, typename _Up>
  constexpr bool **operator!=** (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)
- constexpr basic_string_view< char > **operator""sv** (const char *__str, size_t __len) noexcept
- constexpr basic_string_view< char16_t > **operator""sv** (const char16_t *__str, size_t __len) noexcept
- constexpr basic_string_view< char32_t > **operator""sv** (const char32_t *__str, size_t __len) noexcept
- constexpr basic_string_view< wchar_t > **operator""sv** (const wchar_t *__str, size_t __len) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **operator<** (__type_identity_t< basic_string_view< _CharT, _Traits > > __x, basic_string_view< _CharT, _Traits > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **operator<** (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **operator<** (basic_string_view< _CharT, _Traits > __x, basic_string_view< _CharT, _Traits > __y) noexcept
- template<typename _Tp, typename _Up>
  constexpr bool **operator<** (const _Tp &__t, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **operator<** (const propagate_const< _Tp > &__pt, const _Up &__u)
- template<typename _Tp, typename _Up>
  constexpr bool **operator<** (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)
- template<typename _Tp>
  bool **operator<** (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp1, typename _Tp2>
  bool **operator<** (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept
- template<typename _Tp>
  bool **operator<** (nullptr_t, const shared_ptr< _Tp > &__a) noexcept
- template<typename _Tp, typename _Up>
  constexpr bool **operator<** (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator<<** (basic_ostream< _CharT, _Traits > &__os, basic_string_view< _CharT, _Traits > __str)
- template<typename _Ch, typename _Tr, typename _Tp>
  std::basic_ostream< _Ch, _Tr > & **operator<<** (std::basic_ostream< _Ch, _Tr > &__os, const shared_ptr< _Tp > &__p)
- template<typename _CharT, typename _Traits>
  constexpr bool **operator<=** (__type_identity_t< basic_string_view< _CharT, _Traits > > __x, basic_string_view< _CharT, _Traits > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **operator<=** (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept

- template<typename _CharT, typename _Traits>
  constexpr bool **operator<=** (basic_string_view< _CharT, _Traits > __x, basic_string_view< _CharT, _Traits > __y) noexcept
- template<typename _Tp, typename _Up>
  constexpr bool **operator<=** (const _Tp &__t, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **operator<=** (const propagate_const< _Tp > &__pt, const _Up &__u)
- template<typename _Tp, typename _Up>
  constexpr bool **operator<=** (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)
- template<typename _Tp>
  bool **operator<=** (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp1, typename _Tp2>
  bool **operator<=** (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept
- template<typename _Tp>
  bool **operator<=** (nullptr_t, const shared_ptr< _Tp > &__a) noexcept
- template<typename _Tp, typename _Up>
  constexpr bool **operator<=** (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)
- template<typename _CharT, typename _Traits>
  constexpr bool **operator==** (__type_identity_t< basic_string_view< _CharT, _Traits > > __x, basic_string_view< _CharT, _Traits > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **operator==** (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **operator==** (basic_string_view< _CharT, _Traits > __x, basic_string_view< _CharT, _Traits > __y) noexcept
- template<typename _Tp, typename _Up>
  constexpr bool **operator==** (const _Tp &__t, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **operator==** (const propagate_const< _Tp > &__pt, const _Up &__u)
- template<typename _Tp, typename _Up>
  constexpr bool **operator==** (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)
- template<typename _Tp>
  constexpr bool **operator==** (const propagate_const< _Tp > &__pt, nullptr_t)
- template<typename _Tp>
  bool **operator==** (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp1, typename _Tp2>
  bool **operator==** (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept
- template<typename _Tp>
  constexpr bool **operator==** (nullptr_t, const propagate_const< _Tp > &__pu)
- template<typename _Tp>
  bool **operator==** (nullptr_t, const shared_ptr< _Tp > &__a) noexcept
- template<typename _Tp>
  constexpr bool **operator==** (nullptr_t, observer_ptr< _Tp > __p) noexcept
- template<typename _Tp>
  constexpr bool **operator==** (observer_ptr< _Tp > __p, nullptr_t) noexcept
- template<typename _Tp, typename _Up>
  constexpr bool **operator==** (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)
- template<typename _CharT, typename _Traits>
  constexpr bool **operator>** (__type_identity_t< basic_string_view< _CharT, _Traits > > __x, basic_string_view< _CharT, _Traits > __y) noexcept

- template<typename _CharT, typename _Traits>
  constexpr bool **operator>** (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **operator>** (basic_string_view< _CharT, _Traits > __x, basic_string_view< _CharT, _Traits > __y) noexcept
- template<typename _Tp, typename _Up>
  constexpr bool **operator>** (const _Tp &__t, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **operator>** (const propagate_const< _Tp > &__pt, const _Up &__u)
- template<typename _Tp, typename _Up>
  constexpr bool **operator>** (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)
- template<typename _Tp>
  bool **operator>** (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp1, typename _Tp2>
  bool **operator>** (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept
- template<typename _Tp>
  bool **operator>** (nullptr_t, const shared_ptr< _Tp > &__a) noexcept
- template<typename _Tp, typename _Up>
  constexpr bool **operator>** (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)
- template<typename _CharT, typename _Traits>
  constexpr bool **operator>=** (__type_identity_t< basic_string_view< _CharT, _Traits > > __x, basic_string_view< _CharT, _Traits > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **operator>=** (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **operator>=** (basic_string_view< _CharT, _Traits > __x, basic_string_view< _CharT, _Traits > __y) noexcept
- template<typename _Tp, typename _Up>
  constexpr bool **operator>=** (const _Tp &__t, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **operator>=** (const propagate_const< _Tp > &__pt, const _Up &__u)
- template<typename _Tp, typename _Up>
  constexpr bool **operator>=** (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)
- template<typename _Tp>
  bool **operator>=** (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp1, typename _Tp2>
  bool **operator>=** (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept
- template<typename _Tp>
  bool **operator>=** (nullptr_t, const shared_ptr< _Tp > &__a) noexcept
- template<typename _Tp, typename _Up>
  constexpr bool **operator>=** (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)
- template<typename _IntType>
  _IntType **randint** (_IntType __a, _IntType __b)
- template<typename _Tp, typename _Tp1>
  shared_ptr< _Tp > **reinterpret_pointer_cast** (const shared_ptr< _Tp1 > &__r) noexcept
- void **reseed** ()
- void **reseed** (default_random_engine::result_type __value)
- template<typename _PopulationIterator, typename _SampleIterator, typename _Distance>
  _SampleIterator **sample** (_PopulationIterator __first, _PopulationIterator __last, _SampleIterator __out, _↩
  Distance __n)

- template<typename _PopulationIterator, typename _SampleIterator, typename _Distance, typename _UniformRandomNumberGenerator>
  _SampleIterator sample (_PopulationIterator __first, _PopulationIterator __last, _SampleIterator __out, _Distance __n, _UniformRandomNumberGenerator &&__g)
- template<typename _ForwardIterator, typename _Searcher>
  _ForwardIterator **search** (_ForwardIterator __first, _ForwardIterator __last, const _Searcher &__searcher)
- template<typename _RandomAccessIterator>
  void **shuffle** (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _Tp, typename _Tp1>
  shared_ptr< _Tp > **static_pointer_cast** (const shared_ptr< _Tp1 > &__r) noexcept
- void swap (any &__x, any &__y) noexcept
- template<typename _Tp>
  constexpr void **swap** (observer_ptr< _Tp > &__p1, observer_ptr< _Tp > &__p2) noexcept
- template<typename _Tp>
  constexpr enable_if_t< __is_swappable< _Tp >::value, void > **swap** (propagate_const< _Tp > &__pt, propagate_const< _Tp > &__pt2) noexcept(__is_nothrow_swappable< _Tp >::value)
- template<typename _Tp>
  void **swap** (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b) noexcept
- template<typename _Tp>
  void **swap** (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b) noexcept
- template<typename _Tp, size_t _Nm>
  constexpr array< remove_cv_t< _Tp >, _Nm > to_array (_Tp(&__a)[_Nm]) noexcept(is_nothrow_constructible< remove_cv_t< _Tp >, _Tp & >::value)

- template<typename _ValueType>
  _ValueType any_cast (any &__any)
- template<typename _ValueType, typename enable_if<!is_move_constructible< _ValueType >::value||is_lvalue_reference< _ValueType >::value, bool >::type = true>
  _ValueType any_cast (any &&__any)
- template<typename _ValueType, typename enable_if< is_move_constructible< _ValueType >::value &&!is_lvalue_reference< _ValueType >::value, bool >::type = false>
  _ValueType any_cast (any &&__any)

- template<typename _ValueType>
  const _ValueType * any_cast (const any *__any) noexcept
- template<typename _ValueType>
  _ValueType * any_cast (any *__any) noexcept

Variables

- template<typename _Yp, typename _Tp>
  constexpr bool **__sp_compatible_v**
- template<typename _Tp, typename _Yp>
  constexpr bool **__sp_is_constructible_v**
- constexpr in_place_t in_place
- template<typename _Tp>
  constexpr bool is_bind_expression_v
- template<typename _Tp>
  constexpr bool **is_error_code_enum_v**
- template<typename _Tp>
  constexpr bool **is_error_condition_enum_v**

- template<typename _Tp>
  constexpr int is_placeholder_v
- constexpr nullopt_t nullopt
- template<typename _R1, typename _R2>
  constexpr bool **ratio_equal_v**
- template<typename _R1, typename _R2>
  constexpr bool **ratio_greater_equal_v**
- template<typename _R1, typename _R2>
  constexpr bool **ratio_greater_v**
- template<typename _R1, typename _R2>
  constexpr bool **ratio_less_equal_v**
- template<typename _R1, typename _R2>
  constexpr bool **ratio_less_v**
- template<typename _R1, typename _R2>
  constexpr bool **ratio_not_equal_v**
- template<typename _Tp>
  constexpr size_t **tuple_size_v**

- template<typename _Tp>
  constexpr bool **is_void_v**
- template<typename _Tp>
  constexpr bool **is_null_pointer_v**
- template<typename _Tp>
  constexpr bool **is_integral_v**
- template<typename _Tp>
  constexpr bool **is_floating_point_v**
- template<typename _Tp>
  constexpr bool **is_array_v**
- template<typename _Tp>
  constexpr bool **is_pointer_v**
- template<typename _Tp>
  constexpr bool **is_lvalue_reference_v**
- template<typename _Tp>
  constexpr bool **is_rvalue_reference_v**
- template<typename _Tp>
  constexpr bool **is_member_object_pointer_v**
- template<typename _Tp>
  constexpr bool **is_member_function_pointer_v**
- template<typename _Tp>
  constexpr bool **is_enum_v**
- template<typename _Tp>
  constexpr bool **is_union_v**
- template<typename _Tp>
  constexpr bool **is_class_v**
- template<typename _Tp>
  constexpr bool **is_function_v**
- template<typename _Tp>
  constexpr bool **is_reference_v**
- template<typename _Tp>
  constexpr bool **is_arithmetic_v**
- template<typename _Tp>
  constexpr bool **is_fundamental_v**

- template<typename _Tp>
  constexpr bool **is_object_v**
- template<typename _Tp>
  constexpr bool **is_scalar_v**
- template<typename _Tp>
  constexpr bool **is_compound_v**
- template<typename _Tp>
  constexpr bool **is_member_pointer_v**
- template<typename _Tp>
  constexpr bool **is_const_v**
- template<typename _Tp>
  constexpr bool **is_volatile_v**
- template<typename _Tp>
  constexpr bool **is_trivial_v**
- template<typename _Tp>
  constexpr bool **is_trivially_copyable_v**
- template<typename _Tp>
  constexpr bool **is_standard_layout_v**
- template<typename _Tp>
  constexpr bool **is_pod_v**
- template<typename _Tp>
  constexpr bool **is_literal_type_v**
- template<typename _Tp>
  constexpr bool **is_empty_v**
- template<typename _Tp>
  constexpr bool **is_polymorphic_v**
- template<typename _Tp>
  constexpr bool **is_abstract_v**
- template<typename _Tp>
  constexpr bool **is_final_v**
- template<typename _Tp>
  constexpr bool **is_signed_v**
- template<typename _Tp>
  constexpr bool **is_unsigned_v**
- template<typename _Tp, typename... _Args>
  constexpr bool **is_constructible_v**
- template<typename _Tp>
  constexpr bool **is_default_constructible_v**
- template<typename _Tp>
  constexpr bool **is_copy_constructible_v**
- template<typename _Tp>
  constexpr bool **is_move_constructible_v**
- template<typename _Tp, typename _Up>
  constexpr bool **is_assignable_v**
- template<typename _Tp>
  constexpr bool **is_copy_assignable_v**
- template<typename _Tp>
  constexpr bool **is_move_assignable_v**
- template<typename _Tp>
  constexpr bool **is_destructible_v**
- template<typename _Tp, typename... _Args>
  constexpr bool **is_trivially_constructible_v**

- template<typename _Tp>
  constexpr bool **is_trivially_default_constructible_v**
- template<typename _Tp>
  constexpr bool **is_trivially_copy_constructible_v**
- template<typename _Tp>
  constexpr bool **is_trivially_move_constructible_v**
- template<typename _Tp, typename _Up>
  constexpr bool **is_trivially_assignable_v**
- template<typename _Tp>
  constexpr bool **is_trivially_copy_assignable_v**
- template<typename _Tp>
  constexpr bool **is_trivially_move_assignable_v**
- template<typename _Tp>
  constexpr bool **is_trivially_destructible_v**
- template<typename _Tp, typename... _Args>
  constexpr bool **is_nothrow_constructible_v**
- template<typename _Tp>
  constexpr bool **is_nothrow_default_constructible_v**
- template<typename _Tp>
  constexpr bool **is_nothrow_copy_constructible_v**
- template<typename _Tp>
  constexpr bool **is_nothrow_move_constructible_v**
- template<typename _Tp, typename _Up>
  constexpr bool **is_nothrow_assignable_v**
- template<typename _Tp>
  constexpr bool **is_nothrow_copy_assignable_v**
- template<typename _Tp>
  constexpr bool **is_nothrow_move_assignable_v**
- template<typename _Tp>
  constexpr bool **is_nothrow_destructible_v**
- template<typename _Tp>
  constexpr bool **has_virtual_destructor_v**
- template<typename _Tp>
  constexpr size_t **alignment_of_v**
- template<typename _Tp>
  constexpr size_t **rank_v**
- template<typename _Tp, unsigned _Idx = 0>
  constexpr size_t **extent_v**
- template<typename _Tp, typename _Up>
  constexpr bool **is_same_v**
- template<typename _Tp>
  constexpr bool **is_same_v**< **_Tp, _Tp** >
- template<typename _Base, typename _Derived>
  constexpr bool **is_base_of_v**
- template<typename _From, typename _To>
  constexpr bool **is_convertible_v**

- template<typename... _Bn>
  constexpr bool **conjunction_v**
- template<typename... _Bn>
  constexpr bool **disjunction_v**

- template<typename _Pp>
  constexpr bool **negation_v**

- template<typename...>
  using void_t
- template<typename _Default, template< typename... > class _Op, typename... _Args>
  using detected_or
- template<typename _Default, template< typename... > class _Op, typename... _Args>
  using detected_or_t
- template<template< typename... > class _Op, typename... _Args>
  using detected_t
- template<template< typename... > class _Op, typename... _Args>
  using is_detected
- template<typename _Expected, template< typename... > class _Op, typename... _Args>
  using is_detected_exact
- template<typename _To, template< typename... > class _Op, typename... _Args>
  using is_detected_convertible
- template<template< typename... > class _Op, typename... _Args>
  constexpr bool is_detected_v
- template<typename _Expected, template< typename... > class _Op, typename... _Args>
  constexpr bool is_detected_exact_v
- template<typename _To, template< typename... > class _Op, typename... _Args>
  constexpr bool is_detected_convertible_v

4.16.1 Detailed Description

Namespace for features defined in ISO Technical Specifications.

4.16.2 Function Documentation

gcd()

```
template<typename _Mn, typename _Nn>
common_type_t< _Mn, _Nn > std::experimental::fundamentals_v2::gcd (
            _Mn __m,
            _Nn __n)  [constexpr], [noexcept]
```

Greatest common divisor.

get_deleter()

```
template<typename _Del, typename _Tp>
_Del * std::experimental::fundamentals_v2::get_deleter (
            const shared_ptr< _Tp > & __p)  [inline], [noexcept]
```

C++14 20.8.2.2.10.

lcm()

```
template<typename _Mn, typename _Nn>
common_type_t< _Mn, _Nn > std::experimental::fundamentals_v2::lcm (
            _Mn __m,
            _Nn __n)  [constexpr]
```

Least common multiple.

make_boyer_moore_horspool_searcher()

```
template<typename _RAIter, typename _Hash = std::hash<typename std::iterator_traits<_RAIter>↩
::value_type>, typename _BinaryPredicate = equal_to<>>
boyer_moore_horspool_searcher< _RAIter, _Hash, _BinaryPredicate > std::experimental::fundamentals↩
_v1::make_boyer_moore_horspool_searcher (
            _RAIter __pat_first,
            _RAIter __pat_last,
            _Hash __hf = _Hash(),
            _BinaryPredicate __pred = _BinaryPredicate())  [inline]
```

Generator function for boyer_moore_horspool_searcher.

make_boyer_moore_searcher()

```
template<typename _RAIter, typename _Hash = std::hash<typename std::iterator_traits<_RAIter>↩
::value_type>, typename _BinaryPredicate = equal_to<>>
boyer_moore_searcher< _RAIter, _Hash, _BinaryPredicate > std::experimental::fundamentals_v1↩
::make_boyer_moore_searcher (
            _RAIter __pat_first,
            _RAIter __pat_last,
            _Hash __hf = _Hash(),
            _BinaryPredicate __pred = _BinaryPredicate())  [inline]
```

Generator function for boyer_moore_searcher.

make_default_searcher()

```
template<typename _ForwardIterator, typename _BinaryPredicate = std::equal_to<>>
default_searcher< _ForwardIterator, _BinaryPredicate > std::experimental::fundamentals_v1::make↩
_default_searcher (
            _ForwardIterator __pat_first,
            _ForwardIterator __pat_last,
            _BinaryPredicate __pred = _BinaryPredicate())  [inline]
```

Generator function for default_searcher.

make_ostream_joiner()

```
template<typename _CharT, typename _Traits, typename _DelimT>
ostream_joiner< decay_t< _DelimT >, _CharT, _Traits > std::experimental::fundamentals_v2::make↩
_ostream_joiner (
            basic_ostream< _CharT, _Traits > & __os,
            _DelimT && __delimiter)  [inline]
```

Object generator for ostream_joiner.

not_fn()

```
template<typename _Fn>
auto std::experimental::fundamentals_v2::not_fn (
            _Fn && __fn)  [inline], [noexcept]
```

[func.not_fn] Function template not_fn

sample()

```
template<typename _PopulationIterator, typename _SampleIterator, typename _Distance, typename _↩
UniformRandomNumberGenerator>
_SampleIterator std::experimental::fundamentals_v2::sample (
```

```
        _PopulationIterator __first,
        _PopulationIterator __last,
        _SampleIterator __out,
        _Distance __n,
        _UniformRandomNumberGenerator && __g)
```

Take a random sample from a population.

4.16.3 Variable Documentation

is_bind_expression_v

```
template<typename _Tp>
bool std::experimental::fundamentals_v1::is_bind_expression_v  [constexpr]
```

Variable template for std::is_bind_expression.

is_placeholder_v

```
template<typename _Tp>
int std::experimental::fundamentals_v1::is_placeholder_v  [constexpr]
```

Variable template for std::is_placeholder.

4.17 std::filesystem Namespace Reference

Classes

- class directory_entry
- class directory_iterator
- class file_status
- class filesystem_error
- class path
- class recursive_directory_iterator
- struct space_info

Typedefs

- using file_time_type

Enumerations

- enum class copy_options : unsigned short {
  none , **skip_existing** , **overwrite_existing** , **update_existing** ,
  recursive , **copy_symlinks** , **skip_symlinks** , **directories_only** ,
  create_symlinks , **create_hard_links** }
- enum class directory_options : unsigned char { **none** , **follow_directory_symlink** , **skip_permission_denied** }
- enum class file_type : signed char {
  none , **not_found** , **regular** , **directory** ,
  symlink , **block** , **character** , **fifo** ,
  socket , **unknown** }
- enum class perm_options : unsigned { **replace** , **add** , **remove** , **nofollow** }
- enum class perms : unsigned {
  none , **owner_read** , **owner_write** , **owner_exec** ,
  owner_all , **group_read** , **group_write** , **group_exec** ,
  group_all , **others_read** , **others_write** , **others_exec** ,
  others_all , **all** , **set_uid** , **set_gid** ,
  sticky_bit , **mask** , **unknown** }

Functions

- path **absolute** (const path &__p)
- path **absolute** (const path &__p, error_code &__ec)
- path **canonical** (const path &__p)
- path **canonical** (const path &__p, error_code &__ec)
- void **copy** (const path &__from, const path &__to)
- void **copy** (const path &__from, const path &__to, copy_options __options)
- void **copy** (const path &__from, const path &__to, copy_options __options, error_code &)
- void **copy** (const path &__from, const path &__to, error_code &__ec)
- bool **copy_file** (const path &__from, const path &__to)
- bool **copy_file** (const path &__from, const path &__to, copy_options __option)
- bool **copy_file** (const path &__from, const path &__to, copy_options __option, error_code &)
- bool **copy_file** (const path &__from, const path &__to, error_code &__ec)
- void **copy_symlink** (const path &__existing_symlink, const path &__new_symlink)
- void **copy_symlink** (const path &__existing_symlink, const path &__new_symlink, error_code &__ec) noexcept
- bool **create_directories** (const path &__p)
- bool **create_directories** (const path &__p, error_code &__ec)
- bool **create_directory** (const path &__p)
- bool **create_directory** (const path &__p, const path &__attributes)
- bool **create_directory** (const path &__p, const path &__attributes, error_code &__ec) noexcept
- bool **create_directory** (const path &__p, error_code &__ec) noexcept
- void **create_directory_symlink** (const path &__to, const path &__new_symlink)
- void **create_directory_symlink** (const path &__to, const path &__new_symlink, error_code &__ec) noexcept
- void **create_hard_link** (const path &__to, const path &__new_hard_link)
- void **create_hard_link** (const path &__to, const path &__new_hard_link, error_code &__ec) noexcept
- void **create_symlink** (const path &__to, const path &__new_symlink)
- void **create_symlink** (const path &__to, const path &__new_symlink, error_code &__ec) noexcept
- path **current_path** ()
- void **current_path** (const path &__p)
- void **current_path** (const path &__p, error_code &__ec) noexcept
- path **current_path** (error_code &__ec)
- bool **equivalent** (const path &__p1, const path &__p2)
- bool **equivalent** (const path &__p1, const path &__p2, error_code &__ec) noexcept
- bool **exists** (const path &__p)
- bool **exists** (const path &__p, error_code &__ec) noexcept
- bool **exists** (file_status) noexcept
- uintmax_t **file_size** (const path &)
- uintmax_t **file_size** (const path &, error_code &) noexcept
- uintmax_t **hard_link_count** (const path &)
- uintmax_t **hard_link_count** (const path &, error_code &) noexcept
- size_t **hash_value** (const path &__p) noexcept
- bool **is_block_file** (const path &__p)
- bool **is_block_file** (const path &__p, error_code &__ec) noexcept
- bool **is_block_file** (file_status __s) noexcept
- bool **is_character_file** (const path &__p)
- bool **is_character_file** (const path &__p, error_code &__ec) noexcept
- bool **is_character_file** (file_status __s) noexcept
- bool **is_directory** (const path &__p)
- bool **is_directory** (const path &__p, error_code &__ec) noexcept
- bool **is_directory** (file_status __s) noexcept

- bool **is_empty** (const path &__p)
- bool **is_empty** (const path &__p, error_code &__ec)
- bool **is_fifo** (const path &__p)
- bool **is_fifo** (const path &__p, error_code &__ec) noexcept
- bool **is_fifo** (file_status __s) noexcept
- bool **is_other** (const path &__p)
- bool **is_other** (const path &__p, error_code &__ec) noexcept
- bool **is_other** (file_status) noexcept
- bool **is_regular_file** (const path &__p)
- bool **is_regular_file** (const path &__p, error_code &__ec) noexcept
- bool **is_regular_file** (file_status) noexcept
- bool **is_socket** (const path &__p)
- bool **is_socket** (const path &__p, error_code &__ec) noexcept
- bool **is_socket** (file_status __s) noexcept
- bool **is_symlink** (const path &__p)
- bool **is_symlink** (const path &__p, error_code &__ec) noexcept
- bool **is_symlink** (file_status) noexcept
- file_time_type **last_write_time** (const path &)
- file_time_type **last_write_time** (const path &, error_code &) noexcept
- void **last_write_time** (const path &__p, file_time_type __new_time)
- void **last_write_time** (const path &__p, file_time_type __new_time, error_code &__ec) noexcept
- copy_options & **operator&=** (copy_options &__x, copy_options __y) noexcept
- constexpr copy_options **operator^** (copy_options __x, copy_options __y) noexcept
- copy_options & **operator^=** (copy_options &__x, copy_options __y) noexcept
- constexpr copy_options **operator|** (copy_options __x, copy_options __y) noexcept
- copy_options & **operator|=** (copy_options &__x, copy_options __y) noexcept
- constexpr copy_options **operator~** (copy_options __x) noexcept
- void **permissions** (const path &, perms, perm_options, error_code &) noexcept
- void **permissions** (const path &__p, perms __prms, error_code &__ec) noexcept
- void **permissions** (const path &__p, perms __prms, perm_options __opts=perm_options::replace)
- path **proximate** (const path &__p, const path &__base, error_code &__ec)
- path **proximate** (const path &__p, const path &__base=current_path())
- path **proximate** (const path &__p, error_code &__ec)
- path **read_symlink** (const path &__p)
- path **read_symlink** (const path &__p, error_code &__ec)
- path **relative** (const path &__p, const path &__base, error_code &__ec)
- path **relative** (const path &__p, const path &__base=current_path())
- path **relative** (const path &__p, error_code &__ec)
- bool **remove** (const path &, error_code &) noexcept
- bool **remove** (const path &__p)
- uintmax_t **remove_all** (const path &)
- uintmax_t **remove_all** (const path &, error_code &)
- void **rename** (const path &__from, const path &__to)
- void **rename** (const path &__from, const path &__to, error_code &__ec) noexcept
- void **resize_file** (const path &__p, uintmax_t __size)
- void **resize_file** (const path &__p, uintmax_t __size, error_code &__ec) noexcept
- space_info **space** (const path &__p)
- space_info **space** (const path &__p, error_code &__ec) noexcept
- file_status **status** (const path &)
- file_status **status** (const path &, error_code &) noexcept
- bool **status_known** (file_status) noexcept

- void **swap** (path &__lhs, path &__rhs) noexcept
- file_status **symlink_status** (const path &)
- file_status **symlink_status** (const path &, error_code &) noexcept
- path **temp_directory_path** ()
- path **temp_directory_path** (error_code &__ec)
- template<typename _InputIterator, typename _Require = __detail::_Path2<_InputIterator>, typename _CharT = __detail::__value_type_↩ is_char_or_char8_t<_InputIterator>>
  path u8path (_InputIterator __first, _InputIterator __last)
- template<typename _Source, typename _Require = __detail::_Path<_Source>, typename _CharT = __detail::__value_type_is_char_or↩ _char8_t<_Source>>
  path u8path (const _Source &__source)
- path **weakly_canonical** (const path &__p)
- path **weakly_canonical** (const path &__p, error_code &__ec)

- constexpr perms **operator|** (perms __x, perms __y) noexcept
- constexpr perms **operator^** (perms __x, perms __y) noexcept
- constexpr perms **operator~** (perms __x) noexcept
- perms & **operator&=** (perms &__x, perms __y) noexcept
- perms & **operator|=** (perms &__x, perms __y) noexcept
- perms & **operator^=** (perms &__x, perms __y) noexcept

- constexpr perm_options **operator|** (perm_options __x, perm_options __y) noexcept
- constexpr perm_options **operator^** (perm_options __x, perm_options __y) noexcept
- constexpr perm_options **operator~** (perm_options __x) noexcept
- perm_options & **operator&=** (perm_options &__x, perm_options __y) noexcept
- perm_options & **operator|=** (perm_options &__x, perm_options __y) noexcept
- perm_options & **operator^=** (perm_options &__x, perm_options __y) noexcept

- constexpr directory_options **operator|** (directory_options __x, directory_options __y) noexcept
- constexpr directory_options **operator^** (directory_options __x, directory_options __y) noexcept
- constexpr directory_options **operator~** (directory_options __x) noexcept
- directory_options & **operator&=** (directory_options &__x, directory_options __y) noexcept
- directory_options & **operator|=** (directory_options &__x, directory_options __y) noexcept
- directory_options & **operator^=** (directory_options &__x, directory_options __y) noexcept

4.17.1 Detailed Description

ISO C++ 2017 namespace for File System library.

4.18 std::literals Namespace Reference

Namespaces

- namespace chrono_literals

Functions

- constexpr chrono::day operator""d (unsigned long long __d) noexcept
- template<char... _Digits>
  constexpr chrono::hours operator""h ()
- constexpr chrono::duration< long double, ratio< 3600, 1 > > operator""h (long double __hours)
- template<char... _Digits>
  constexpr chrono::minutes operator""min ()
- constexpr chrono::duration< long double, ratio< 60, 1 > > operator""min (long double __mins)
- template<char... _Digits>
  constexpr chrono::milliseconds operator""ms ()
- constexpr chrono::duration< long double, milli > operator""ms (long double __msecs)
- template<char... _Digits>
  constexpr chrono::nanoseconds operator""ns ()
- constexpr chrono::duration< long double, nano > operator""ns (long double __nsecs)
- template<char... _Digits>
  constexpr chrono::seconds operator""s ()
- constexpr chrono::duration< long double > operator""s (long double __secs)
- constexpr basic_string_view< char > **operator""sv** (const char *__str, size_t __len) noexcept
- constexpr basic_string_view< char16_t > **operator""sv** (const char16_t *__str, size_t __len) noexcept
- constexpr basic_string_view< char32_t > **operator""sv** (const char32_t *__str, size_t __len) noexcept
- constexpr basic_string_view< wchar_t > **operator""sv** (const wchar_t *__str, size_t __len) noexcept
- template<char... _Digits>
  constexpr chrono::microseconds operator""us ()
- constexpr chrono::duration< long double, micro > operator""us (long double __usecs)
- constexpr chrono::year operator""y (unsigned long long __y) noexcept

4.18.1 Detailed Description

ISO C++ inline namespace for literal suffixes.

4.19 std::literals::chrono_literals Namespace Reference

Functions

- constexpr chrono::day operator""d (unsigned long long __d) noexcept
- template<char... _Digits>
  constexpr chrono::hours operator""h ()
- constexpr chrono::duration< long double, ratio< 3600, 1 > > operator""h (long double __hours)
- template<char... _Digits>
  constexpr chrono::minutes operator""min ()
- constexpr chrono::duration< long double, ratio< 60, 1 > > operator""min (long double __mins)
- template<char... _Digits>
  constexpr chrono::milliseconds operator""ms ()
- constexpr chrono::duration< long double, milli > operator""ms (long double __msecs)
- template<char... _Digits>
  constexpr chrono::nanoseconds operator""ns ()
- constexpr chrono::duration< long double, nano > operator""ns (long double __nsecs)
- template<char... _Digits>
  constexpr chrono::seconds operator""s ()
- constexpr chrono::duration< long double > operator""s (long double __secs)
- template<char... _Digits>
  constexpr chrono::microseconds operator""us ()
- constexpr chrono::duration< long double, micro > operator""us (long double __usecs)
- constexpr chrono::year operator""y (unsigned long long __y) noexcept

4.19.1 Detailed Description

ISO C++ 2014 namespace for suffixes for duration literals.
These suffixes can be used to create `chrono::duration` values with tick periods of hours, minutes, seconds, milliseconds, microseconds or nanoseconds. For example, `std::chrono::seconds(5)` can be written as `5s` after making the suffix visible in the current scope. The suffixes can be made visible by a using-directive or using-declaration such as:

- `using namespace std::chrono_literals;`
- `using namespace std::literals;`
- `using namespace std::chrono;`
- `using namespace std;`
- `using std::chrono_literals::operator""s;`

The result of these suffixes on an integer literal is one of the standard typedefs such as `std::chrono::hours`. The result on a floating-point literal is a duration type with the specified tick period and an unspecified floating-point representation, for example `1.5e2ms` might be equivalent to `chrono::duration<long double, chrono::milli>(1.5e2)`.

Since

C+14

4.20 std::placeholders Namespace Reference

Variables

- const _Placeholder< 1 > **_1**
- const _Placeholder< 10 > **_10**
- const _Placeholder< 11 > **_11**
- const _Placeholder< 12 > **_12**
- const _Placeholder< 13 > **_13**
- const _Placeholder< 14 > **_14**
- const _Placeholder< 15 > **_15**
- const _Placeholder< 16 > **_16**
- const _Placeholder< 17 > **_17**
- const _Placeholder< 18 > **_18**
- const _Placeholder< 19 > **_19**
- const _Placeholder< 2 > **_2**
- const _Placeholder< 20 > **_20**
- const _Placeholder< 21 > **_21**
- const _Placeholder< 22 > **_22**
- const _Placeholder< 23 > **_23**
- const _Placeholder< 24 > **_24**
- const _Placeholder< 25 > **_25**
- const _Placeholder< 26 > **_26**
- const _Placeholder< 27 > **_27**
- const _Placeholder< 28 > **_28**
- const _Placeholder< 29 > **_29**
- const _Placeholder< 3 > **_3**
- const _Placeholder< 4 > **_4**
- const _Placeholder< 5 > **_5**
- const _Placeholder< 6 > **_6**
- const _Placeholder< 7 > **_7**
- const _Placeholder< 8 > **_8**
- const _Placeholder< 9 > **_9**

4.20.1 Detailed Description

ISO C++ 2011 namespace for std::bind placeholders.

Since

C++11

4.21 std::regex_constants Namespace Reference

5.1 Regular Expression Syntax Options

- enum syntax_option_type : unsigned int {
  _S_icase , **_S_nosubs** , **_S_optimize** , **_S_collate** ,
  _S_ECMAScript , **_S_basic** , **_S_extended** , **_S_awk** ,
  _S_grep , **_S_egrep** , **_S_polynomial** , **_S_multiline** }
- constexpr syntax_option_type icase
- constexpr syntax_option_type nosubs
- constexpr syntax_option_type optimize
- constexpr syntax_option_type collate
- constexpr syntax_option_type ECMAScript
- constexpr syntax_option_type basic
- constexpr syntax_option_type extended
- constexpr syntax_option_type awk
- constexpr syntax_option_type grep
- constexpr syntax_option_type egrep
- constexpr syntax_option_type multiline
- constexpr syntax_option_type __multiline
- constexpr syntax_option_type __polynomial
- constexpr syntax_option_type operator& (syntax_option_type __a, syntax_option_type __b) noexcept
- constexpr syntax_option_type operator| (syntax_option_type __a, syntax_option_type __b) noexcept
- constexpr syntax_option_type operator^ (syntax_option_type __a, syntax_option_type __b) noexcept
- constexpr syntax_option_type operator~ (syntax_option_type __a) noexcept
- constexpr syntax_option_type & operator&= (syntax_option_type &__a, syntax_option_type __b) noexcept
- constexpr syntax_option_type & operator|= (syntax_option_type &__a, syntax_option_type __b) noexcept
- constexpr syntax_option_type & operator^= (syntax_option_type &__a, syntax_option_type __b) noexcept

5.2 Matching Rules

Matching a regular expression against a sequence of characters [first, last) proceeds according to the rules of the grammar specified for the regular expression object, modified according to the effects listed below for any bitmask elements set.

- enum match_flag_type : unsigned int {
  _S_default , **_S_not_bol** , **_S_not_eol** , **_S_not_bow** ,
  _S_not_eow , **_S_any** , **_S_not_null** , **_S_continuous** ,
  _S_prev_avail , **_S_sed** , **_S_no_copy** , **_S_first_only** ,
  _S_match_flag_last }
- constexpr match_flag_type match_default
- constexpr match_flag_type match_not_bol
- constexpr match_flag_type match_not_eol
- constexpr match_flag_type match_not_bow
- constexpr match_flag_type match_not_eow
- constexpr match_flag_type match_any

- constexpr match_flag_type match_not_null
- constexpr match_flag_type match_continuous
- constexpr match_flag_type match_prev_avail
- constexpr match_flag_type format_default
- constexpr match_flag_type format_sed
- constexpr match_flag_type format_no_copy
- constexpr match_flag_type format_first_only
- constexpr match_flag_type operator& (match_flag_type __a, match_flag_type __b) noexcept
- constexpr match_flag_type operator| (match_flag_type __a, match_flag_type __b) noexcept
- constexpr match_flag_type operator^ (match_flag_type __a, match_flag_type __b) noexcept
- constexpr match_flag_type operator~ (match_flag_type __a) noexcept
- constexpr match_flag_type & operator&= (match_flag_type &__a, match_flag_type __b) noexcept
- constexpr match_flag_type & operator|= (match_flag_type &__a, match_flag_type __b) noexcept
- constexpr match_flag_type & operator^= (match_flag_type &__a, match_flag_type __b) noexcept

5.3 Error Types

- enum error_type {
  _S_error_collate , **_S_error_ctype** , **_S_error_escape** , **_S_error_backref** ,
  _S_error_brack , **_S_error_paren** , **_S_error_brace** , **_S_error_badbrace** ,
  _S_error_range , **_S_error_space** , **_S_error_badrepeat** , **_S_error_complexity** ,
  _S_error_stack , **_S_null** , **_S_grammar** }
- constexpr error_type error_collate (_S_error_collate)
- constexpr error_type error_ctype (_S_error_ctype)
- constexpr error_type error_escape (_S_error_escape)
- constexpr error_type error_backref (_S_error_backref)
- constexpr error_type error_brack (_S_error_brack)
- constexpr error_type error_paren (_S_error_paren)
- constexpr error_type error_brace (_S_error_brace)
- constexpr error_type error_badbrace (_S_error_badbrace)
- constexpr error_type error_range (_S_error_range)
- constexpr error_type error_space (_S_error_space)
- constexpr error_type error_badrepeat (_S_error_badrepeat)
- constexpr error_type error_complexity (_S_error_complexity)
- constexpr error_type error_stack (_S_error_stack)

4.21.1 Detailed Description

ISO C++ 2011 namespace for options and flags used with std::regex.

4.21.2 Enumeration Type Documentation

error_type

```
enum std::regex_constants::error_type
```

The expression contained an invalid collating element name.

match_flag_type

```
enum std::regex_constants::match_flag_type :  unsigned int
```

This is a bitmask type indicating regex matching rules.
The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

syntax_option_type

```
enum std::regex_constants::syntax_option_type :  unsigned int
```

This is a bitmask type indicating how to interpret the regex.
The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.
A valid value of type syntax_option_type shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

4.21.3 Function Documentation

error_backref()

```
error_type std::regex_constants::error_backref (
            _S_error_backref )  [inline], [constexpr]
```

The expression contained an invalid back reference.

error_badbrace()

```
error_type std::regex_constants::error_badbrace (
            _S_error_badbrace )  [inline], [constexpr]
```

The expression contained an invalid range in a {} expression.

error_badrepeat()

```
error_type std::regex_constants::error_badrepeat (
            _S_error_badrepeat )  [inline], [constexpr]
```

One of *∗?+{* was not preceded by a valid regular expression.

error_brace()

```
error_type std::regex_constants::error_brace (
            _S_error_brace )  [inline], [constexpr]
```

The expression contained mismatched { and }

error_brack()

```
error_type std::regex_constants::error_brack (
            _S_error_brack )  [inline], [constexpr]
```

The expression contained mismatched [and].

error_collate()

```
error_type std::regex_constants::error_collate (
            _S_error_collate )  [inline], [constexpr]
```

The expression contained an invalid collating element name.

error_complexity()

```
error_type std::regex_constants::error_complexity (
            _S_error_complexity )  [inline], [constexpr]
```

The complexity of an attempted match against a regular expression exceeded a pre-set level.

error_ctype()

```
error_type std::regex_constants::error_ctype (
            _S_error_ctype )  [inline], [constexpr]
```

The expression contained an invalid character class name.

error_escape()

```
error_type std::regex_constants::error_escape (
            _S_error_escape )  [inline], [constexpr]
```

The expression contained an invalid escaped character, or a trailing escape.

error_paren()

```
error_type std::regex_constants::error_paren (
            _S_error_paren )  [inline], [constexpr]
```

The expression contained mismatched (and).

error_range()

```
error_type std::regex_constants::error_range (
            _S_error_range )  [inline], [constexpr]
```

The expression contained an invalid character range, such as [b-a] in most encodings.

error_space()

```
error_type std::regex_constants::error_space (
            _S_error_space )  [inline], [constexpr]
```

There was insufficient memory to convert the expression into a finite state machine.

error_stack()

```
error_type std::regex_constants::error_stack (
            _S_error_stack )  [inline], [constexpr]
```

There was insufficient memory to determine whether the regular expression could match the specified character sequence.

operator&() [1/2]

```
match_flag_type std::regex_constants::operator& (
            match_flag_type __a,
            match_flag_type __b)  [constexpr], [noexcept]
```

This is a bitmask type indicating regex matching rules.
The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

operator&() [2/2]

```
syntax_option_type std::regex_constants::operator& (
            syntax_option_type __a,
            syntax_option_type __b)  [constexpr], [noexcept]
```

This is a bitmask type indicating how to interpret the regex.
The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.
A valid value of type syntax_option_type shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

operator&=() [1/2]

```
match_flag_type & std::regex_constants::operator&= (
            match_flag_type & __a,
            match_flag_type __b)  [inline], [constexpr], [noexcept]
```

This is a bitmask type indicating regex matching rules.
The match_flag_type is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

operator&=() [2/2]

```
syntax_option_type & std::regex_constants::operator&= (
            syntax_option_type & __a,
            syntax_option_type __b)  [inline], [constexpr], [noexcept]
```

This is a bitmask type indicating how to interpret the regex.
The syntax_option_type is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.
A valid value of type syntax_option_type shall have exactly one of the elements ECMAScript, basic, extended, awk, grep, egrep set.

operator^() [1/2]

```
match_flag_type std::regex_constants::operator^ (
            match_flag_type __a,
            match_flag_type __b)  [constexpr], [noexcept]
```

This is a bitmask type indicating regex matching rules.
The match_flag_type is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

operator^() [2/2]

```
syntax_option_type std::regex_constants::operator^ (
            syntax_option_type __a,
            syntax_option_type __b)  [constexpr], [noexcept]
```

This is a bitmask type indicating how to interpret the regex.
The syntax_option_type is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.
A valid value of type syntax_option_type shall have exactly one of the elements ECMAScript, basic, extended, awk, grep, egrep set.

operator^=() [1/2]

```
match_flag_type & std::regex_constants::operator^= (
            match_flag_type & __a,
            match_flag_type __b)  [inline], [constexpr], [noexcept]
```

This is a bitmask type indicating regex matching rules.
The match_flag_type is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

operator^=() [2/2]

```
syntax_option_type & std::regex_constants::operator^= (
            syntax_option_type & __a,
            syntax_option_type __b)  [inline], [constexpr], [noexcept]
```

This is a bitmask type indicating how to interpret the regex.

The syntax_option_type is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.
A valid value of type syntax_option_type shall have exactly one of the elements ECMAScript, basic, extended, awk, grep, egrep set.

operator" | () [1/2]

```
match_flag_type std::regex_constants::operator| (
            match_flag_type __a,
            match_flag_type __b)  [constexpr], [noexcept]
```

This is a bitmask type indicating regex matching rules.
The match_flag_type is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

operator" | () [2/2]

```
syntax_option_type std::regex_constants::operator| (
            syntax_option_type __a,
            syntax_option_type __b)  [constexpr], [noexcept]
```

This is a bitmask type indicating how to interpret the regex.
The syntax_option_type is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.
A valid value of type syntax_option_type shall have exactly one of the elements ECMAScript, basic, extended, awk, grep, egrep set.

operator" |=() [1/2]

```
match_flag_type & std::regex_constants::operator|= (
            match_flag_type & __a,
            match_flag_type __b)  [inline], [constexpr], [noexcept]
```

This is a bitmask type indicating regex matching rules.
The match_flag_type is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

operator" |=() [2/2]

```
syntax_option_type & std::regex_constants::operator|= (
            syntax_option_type & __a,
            syntax_option_type __b)  [inline], [constexpr], [noexcept]
```

This is a bitmask type indicating how to interpret the regex.
The syntax_option_type is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.
A valid value of type syntax_option_type shall have exactly one of the elements ECMAScript, basic, extended, awk, grep, egrep set.

operator~() [1/2]

```
match_flag_type std::regex_constants::operator~ (
            match_flag_type __a)  [constexpr], [noexcept]
```

This is a bitmask type indicating regex matching rules.
The match_flag_type is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

operator∼() [2/2]

```
syntax_option_type std::regex_constants::operator~ (
            syntax_option_type __a)  [constexpr], [noexcept]
```

This is a bitmask type indicating how to interpret the regex.
The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.
A valid value of type syntax_option_type shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

4.21.4 Variable Documentation

__multiline

```
syntax_option_type std::regex_constants::__multiline  [inline], [constexpr]
```

Extension: Equivalent to regex_constants::multiline for C++11 and C++14.

__polynomial

```
syntax_option_type std::regex_constants::__polynomial  [inline], [constexpr]
```

Extension: Ensure both space complexity of compiled regex and time complexity execution are not exponential. If specified in a regex with back-references, the exception regex_constants::error_complexity will be thrown.

awk

```
syntax_option_type std::regex_constants::awk  [inline], [constexpr]
```

Specifies that the grammar recognized by the regular expression engine is that used by POSIX utility awk in IEEE Std 1003.1-2001. This option is identical to syntax_option_type extended, except that C-style escape sequences are supported. These sequences are: \\, \a, \b, \f, \n, \r, \t , \v, \&apos„ &apos„ and \ddd (where ddd is one, two, or three octal digits).

basic

```
syntax_option_type std::regex_constants::basic  [inline], [constexpr]
```

Specifies that the grammar recognized by the regular expression engine is that used by POSIX basic regular expressions in IEEE Std 1003.1-2001, Portable Operating System Interface (POSIX), Base Definitions and Headers, Section 9, Regular Expressions [IEEE, Information Technology – Portable Operating System Interface (POSIX), IEEE Standard 1003.1-2001].

collate

```
syntax_option_type std::regex_constants::collate  [inline], [constexpr]
```

Specifies that character ranges of the form [a-b] should be locale sensitive.

ECMAScript

```
syntax_option_type std::regex_constants::ECMAScript  [inline], [constexpr]
```

Specifies that the grammar recognized by the regular expression engine is that used by ECMAScript in ECMA-262 [Ecma International, ECMAScript Language Specification, Standard Ecma-262, third edition, 1999], as modified in section [28.13]. This grammar is similar to that defined in the PERL scripting language but extended with elements found in the POSIX regular expression grammar.

egrep

```
syntax_option_type std::regex_constants::egrep  [inline], [constexpr]
```

Specifies that the grammar recognized by the regular expression engine is that used by POSIX utility grep when given the -E option in IEEE Std 1003.1-2001. This option is identical to syntax_option_type extended, except that newlines are treated as whitespace.

extended

`syntax_option_type std::regex_constants::extended [inline], [constexpr]`
Specifies that the grammar recognized by the regular expression engine is that used by POSIX extended regular expressions in IEEE Std 1003.1-2001, Portable Operating System Interface (POSIX), Base Definitions and Headers, Section 9, Regular Expressions.

format_default

`match_flag_type std::regex_constants::format_default [inline], [constexpr]`
When a regular expression match is to be replaced by a new string, the new string is constructed using the rules used by the ECMAScript replace function in ECMA- 262 [Ecma International, ECMAScript Language Specification, Standard Ecma-262, third edition, 1999], part 15.5.4.11 String.prototype.replace. In addition, during search and replace operations all non-overlapping occurrences of the regular expression are located and replaced, and sections of the input that did not match the expression are copied unchanged to the output string.
Format strings (from ECMA-262 [15.5.4.11]):

- $$ The dollar-sign itself ($)
- $& The matched substring.
- $‘ The portion of *string* that precedes the matched substring. This would be match_results::prefix().
- $’ The portion of *string* that follows the matched substring. This would be match_results::suffix().
- $n The nth capture, where n is in [1,9] and $n is not followed by a decimal digit. If n <= match_results::size() and the nth capture is undefined, use the empty string instead. If n > match_results::size(), the result is implementation-defined.
- $nn The nnth capture, where nn is a two-digit decimal number on [01, 99]. If nn <= match_results::size() and the nth capture is undefined, use the empty string instead. If nn > match_results::size(), the result is implementation-defined.

Referenced by std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const cha
std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const char * >::format(),
std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const char * >::format(),
and std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const char * >::form

format_first_only

`match_flag_type std::regex_constants::format_first_only [inline], [constexpr]`
When specified during a search and replace operation, only the first occurrence of the regular expression shall be replaced.

format_no_copy

`match_flag_type std::regex_constants::format_no_copy [inline], [constexpr]`
During a search and replace operation, sections of the character container sequence being searched that do not match the regular expression shall not be copied to the output string.

format_sed

match_flag_type std::regex_constants::format_sed [inline], [constexpr]
When a regular expression match is to be replaced by a new string, the new string is constructed using the rules used by the POSIX sed utility in IEEE Std 1003.1- 2001 [IEEE, Information Technology – Portable Operating System Interface (POSIX), IEEE Standard 1003.1-2001].
Referenced by std::regex_traits< _CharT >::lookup_classname().

grep

syntax_option_type std::regex_constants::grep [inline], [constexpr]
Specifies that the grammar recognized by the regular expression engine is that used by POSIX utility grep in IEEE Std 1003.1-2001. This option is identical to syntax_option_type basic, except that newlines are treated as whitespace.

icase

syntax_option_type std::regex_constants::icase [inline], [constexpr]
Specifies that the matching of regular expressions against a character sequence shall be performed without regard to case.

match_any

match_flag_type std::regex_constants::match_any [inline], [constexpr]
If more than one match is possible then any match is an acceptable result.

match_continuous

match_flag_type std::regex_constants::match_continuous [inline], [constexpr]
The expression only matches a sub-sequence that begins at first .
Referenced by std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++().

match_default

match_flag_type std::regex_constants::match_default [inline], [constexpr]
The default matching rules.
Referenced by std::regex_iterator< const char * >::regex_iterator(), std::regex_token_iterator< const char * >::regex_token_iterator(), std::regex_token_iterator< const char * >::regex_token_iterator(), std::regex_token_iterator< const char * >::regex_token_iterator(), std::regex_token_iterator< const char * >::regex_token_iterator(), std::regex_match(), std::regex_match(), std::regex_match(), std::regex_match(), std::regex_match(), std::regex_match(), std::regex_match(), std::regex_replace(), std::regex_replace(), std::regex_replace(), std::regex_replace(), std::regex_replace(), std::regex_replace(), std::regex_search(), std::regex_search(), std::regex_search(), std::regex_search(), std::regex_search(), std::regex_search(), and std::regex_search().

match_not_bol

match_flag_type std::regex_constants::match_not_bol [inline], [constexpr]
The first character in the sequence [first, last) is treated as though it is not at the beginning of a line, so the character (^) in the regular expression shall not match [first, first).

match_not_bow

match_flag_type std::regex_constants::match_not_bow [inline], [constexpr]
The expression \b is not matched against the sub-sequence [first,first).

match_not_eol

match_flag_type std::regex_constants::match_not_eol [inline], [constexpr]
The last character in the sequence [first, last) is treated as though it is not at the end of a line, so the character ($) in the regular expression shall not match [last, last).

match_not_eow

match_flag_type std::regex_constants::match_not_eow [inline], [constexpr]
The expression \b should not be matched against the sub-sequence [last,last).

match_not_null

match_flag_type std::regex_constants::match_not_null [inline], [constexpr]
The expression does not match an empty sequence.
Referenced by std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++().

match_prev_avail

match_flag_type std::regex_constants::match_prev_avail [inline], [constexpr]
--first is a valid iterator position. When this flag is set then the flags match_not_bol and match_not_bow are ignored by the algorithms regex_match, regex_search, and regex_replace, and by the iterators regex_iterator and regex_token_iterator.
Referenced by std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++().

multiline

syntax_option_type std::regex_constants::multiline [inline], [constexpr]
Specifies that the ^ anchor matches at the beginning of a line, and the $ anchor matches at the end of a line, not only at the beginning/end of the input. Valid for the ECMAScript syntax, ignored otherwise.

Since

C++17

nosubs

syntax_option_type std::regex_constants::nosubs [inline], [constexpr]
Specifies that when a regular expression is matched against a character container sequence, no sub-expression matches are to be stored in the supplied match_results structure.

optimize

syntax_option_type std::regex_constants::optimize [inline], [constexpr]
Specifies that the regular expression engine should pay more attention to the speed with which regular expressions are matched, and less to the speed with which regular expression objects are constructed. Otherwise it has no detectable effect on the program output.

4.22 std::rel_ops Namespace Reference

Functions

- template<class _Tp>
  bool operator!= (const _Tp &__x, const _Tp &__y)
- template<class _Tp>
  bool operator<= (const _Tp &__x, const _Tp &__y)

- template<class _Tp>
  bool operator> (const _Tp &__x, const _Tp &__y)
- template<class _Tp>
  bool operator>= (const _Tp &__x, const _Tp &__y)

4.22.1 Detailed Description

The generated relational operators are sequestered here.
Libstdc++ headers must not use the contents of `rel_ops`. User code should also avoid them, because unconstrained function templates are too greedy and can easily cause ambiguities.
C++20 default comparisons are a better solution.

4.22.2 Function Documentation

operator"!=()

```
template<class _Tp>
bool std::rel_ops::operator!= (
            const _Tp & __x,
            const _Tp & __y)  [inline]
```

Defines != for arbitrary types, in terms of ==.

Parameters

_↩ _x	A thing.
_↩ _y	Another thing.

Returns

__x != __y

This function uses == to determine its result.

operator<=()

```
template<class _Tp>
bool std::rel_ops::operator<= (
            const _Tp & __x,
            const _Tp & __y)  [inline]
```

Defines <= for arbitrary types, in terms of <.

Parameters

_↩ _x	A thing.
_↩ _y	Another thing.

Returns

__x <= __y

This function uses < to determine its result.

operator>()

```
template<class _Tp>
bool std::rel_ops::operator> (
            const _Tp & __x,
            const _Tp & __y)  [inline]
```

Defines > for arbitrary types, in terms of <.

Parameters

_↩ _x	A thing.
_↩ _y	Another thing.

Returns

__x > __y

This function uses < to determine its result.

operator>=()

```
template<class _Tp>
bool std::rel_ops::operator>= (
            const _Tp & __x,
            const _Tp & __y)  [inline]
```

Defines >= for arbitrary types, in terms of <.

Parameters

_↩ _x	A thing.
_↩ _y	Another thing.

Returns

__x >= __y

This function uses < to determine its result.

4.23 std::this_thread Namespace Reference

Functions

- thread::id get_id () noexcept
- template<typename _Rep, typename _Period>
  void sleep_for (const chrono::duration< _Rep, _Period > &__rtime)
- template<typename _Clock, typename _Duration>
  void sleep_until (const chrono::time_point< _Clock, _Duration > &__atime)
- void yield () noexcept

4.23.1 Detailed Description

ISO C++ 2011 namespace for interacting with the current thread.
C++11 30.3.2 [thread.thread.this] Namespace this_thread.

4.23.2 Function Documentation

get_id()

```
thread::id std::this_thread::get_id ()  [inline], [noexcept]
```
The unique identifier of the current thread.

sleep_for()

```
template<typename _Rep, typename _Period>
void std::this_thread::sleep_for (
            const chrono::duration< _Rep, _Period > & __rtime)  [inline]
```
this_thread::sleep_for
References std::chrono::duration_cast().
Referenced by sleep_until().

sleep_until()

```
template<typename _Clock, typename _Duration>
void std::this_thread::sleep_until (
            const chrono::time_point< _Clock, _Duration > & __atime)  [inline]
```
this_thread::sleep_until
References sleep_for().

yield()

```
void std::this_thread::yield ()  [inline], [noexcept]
```
Allow the implementation to schedule a different thread.

4.24 std::tr1 Namespace Reference

Namespaces

- namespace __detail

Functions

- template<typename _Tp>
  std::complex< _Tp > **acos** (const std::complex< _Tp > &)
- template<typename _Tp>
  std::complex< _Tp > **acosh** (const std::complex< _Tp > &)
- template<typename _Tp>
  _Tp arg (const complex< _Tp > &)
- template<typename _Tp>
  std::complex< _Tp > **asin** (const std::complex< _Tp > &)
- template<typename _Tp>
  std::complex< _Tp > **asinh** (const std::complex< _Tp > &)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type assoc_laguerre (unsigned int __n, unsigned int __m, _Tp __x)
- float **assoc_laguerref** (unsigned int __n, unsigned int __m, float __x)
- long double **assoc_laguerrel** (unsigned int __n, unsigned int __m, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type assoc_legendre (unsigned int __l, unsigned int __m, _Tp __x)
- float **assoc_legendref** (unsigned int __l, unsigned int __m, float __x)
- long double **assoc_legendrel** (unsigned int __l, unsigned int __m, long double __x)

- template<typename _Tp>
  std::complex< _Tp > **atan** (const std::complex< _Tp > &)
- template<typename _Tp>
  std::complex< _Tp > **atanh** (const std::complex< _Tp > &)
- template<typename _Tpx, typename _Tpy>
  __gnu_cxx::__promote_2< _Tpx, _Tpy >::__type beta (_Tpx __x, _Tpy __y)
- float **betaf** (float __x, float __y)
- long double **betal** (long double __x, long double __y)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type comp_ellint_1 (_Tp __k)
- float **comp_ellint_1f** (float __k)
- long double **comp_ellint_1l** (long double __k)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type comp_ellint_2 (_Tp __k)
- float **comp_ellint_2f** (float __k)
- long double **comp_ellint_2l** (long double __k)
- template<typename _Tp, typename _Tpn>
  __gnu_cxx::__promote_2< _Tp, _Tpn >::__type comp_ellint_3 (_Tp __k, _Tpn __nu)
- float **comp_ellint_3f** (float __k, float __nu)
- long double **comp_ellint_3l** (long double __k, long double __nu)
- template<typename _Tpa, typename _Tpc, typename _Tp>
  __gnu_cxx::__promote_3< _Tpa, _Tpc, _Tp >::__type conf_hyperg (_Tpa __a, _Tpc __c, _Tp __x)
- float conf_hypergf (float __a, float __c, float __x)
- long double conf_hypergl (long double __a, long double __c, long double __x)
- template<typename _Tp>
  std::complex< typename __gnu_cxx::__promote< _Tp >::__type > **conj** (_Tp __x)
- template<typename _Tp>
  std::complex< _Tp > **conj** (const std::complex< _Tp > &__z)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl_bessel_i (_Tpnu __nu, _Tp __x)
- float **cyl_bessel_if** (float __nu, float __x)
- long double **cyl_bessel_il** (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl_bessel_j (_Tpnu __nu, _Tp __x)
- float **cyl_bessel_jf** (float __nu, float __x)
- long double **cyl_bessel_jl** (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl_bessel_k (_Tpnu __nu, _Tp __x)
- float **cyl_bessel_kf** (float __nu, float __x)
- long double **cyl_bessel_kl** (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl_neumann (_Tpnu __nu, _Tp __x)
- float **cyl_neumannf** (float __nu, float __x)
- long double **cyl_neumannl** (long double __nu, long double __x)
- template<typename _Tp, typename _Tpp>
  __gnu_cxx::__promote_2< _Tp, _Tpp >::__type ellint_1 (_Tp __k, _Tpp __phi)
- float **ellint_1f** (float __k, float __phi)
- long double **ellint_1l** (long double __k, long double __phi)
- template<typename _Tp, typename _Tpp>
  __gnu_cxx::__promote_2< _Tp, _Tpp >::__type ellint_2 (_Tp __k, _Tpp __phi)
- float **ellint_2f** (float __k, float __phi)
- long double **ellint_2l** (long double __k, long double __phi)

- template<typename _Tp, typename _Tpn, typename _Tpp>
  __gnu_cxx::__promote_3< _Tp, _Tpn, _Tpp >::__type ellint_3 (_Tp __k, _Tpn __nu, _Tpp __phi)
- float **ellint_3f** (float __k, float __nu, float __phi)
- long double **ellint_3l** (long double __k, long double __nu, long double __phi)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type expint (_Tp __x)
- float **expintf** (float __x)
- long double **expintl** (long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type **fabs** (_Tp __x)
- template<typename _Tp>
  std::complex< _Tp > fabs (const std::complex< _Tp > &__z)
- float **fabs** (float __x)
- long double **fabs** (long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type hermite (unsigned int __n, _Tp __x)
- float **hermitef** (unsigned int __n, float __x)
- long double **hermitel** (unsigned int __n, long double __x)
- template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp>
  __gnu_cxx::__promote_4< _Tpa, _Tpb, _Tpc, _Tp >::__type hyperg (_Tpa __a, _Tpb __b, _Tpc __c, _Tp __x)
- float hypergf (float __a, float __b, float __c, float __x)
- long double hypergl (long double __a, long double __b, long double __c, long double __x)
- template<typename _Tp>
  constexpr _Tp **imag** (const complex< _Tp > &__z)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type laguerre (unsigned int __n, _Tp __x)
- float **laguerref** (unsigned int __n, float __x)
- long double **laguerrel** (unsigned int __n, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type legendre (unsigned int __n, _Tp __x)
- float **legendref** (unsigned int __n, float __x)
- long double **legendrel** (unsigned int __n, long double __x)
- template<typename _Tp>
  _Tp constexpr norm (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > polar (const _Tp &, const _Tp &=_Tp(0))
- template<typename _Tp, typename _Up>
  std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > **polar** (const _Tp &__rho, const _Up &__theta)
- template<typename _Tp, typename _Up>
  __gnu_cxx::__promote_2< _Tp, _Up >::__type **pow** (_Tp __x, _Up __y)
- template<typename _Tp>
  std::complex< _Tp > **pow** (const _Tp &__x, const std::complex< _Tp > &__y)
- template<typename _Tp, typename _Up>
  std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > **pow** (const _Tp &__x, const std::complex< _Up > &__y)
- template<typename _Tp>
  std::complex< _Tp > **pow** (const std::complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp, typename _Up>
  std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > pow (const std::complex< _Tp > &__x, const _Up &__y)

- template<typename _Tp>
  std::complex< _Tp > **pow** (const std::complex< _Tp > &__x, const std::complex< _Tp > &__y)
- template<typename _Tp, typename _Up>
  std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > **pow** (const std::complex< _Tp > &__x, const std::complex< _Up > &__y)
- float **pow** (float __x, float __y)
- long double **pow** (long double __x, long double __y)
- template<typename _Tp>
  constexpr _Tp **real** (const complex< _Tp > &__z)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type riemann_zeta (_Tp __x)
- float **riemann_zetaf** (float __x)
- long double **riemann_zetal** (long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type sph_bessel (unsigned int __n, _Tp __x)
- float **sph_besself** (unsigned int __n, float __x)
- long double **sph_bessell** (unsigned int __n, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type sph_legendre (unsigned int __l, unsigned int __m, _Tp __theta)
- float **sph_legendref** (unsigned int __l, unsigned int __m, float __theta)
- long double **sph_legendrel** (unsigned int __l, unsigned int __m, long double __theta)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type sph_neumann (unsigned int __n, _Tp __x)
- float **sph_neumannf** (unsigned int __n, float __x)
- long double **sph_neumannl** (unsigned int __n, long double __x)

4.24.1 Detailed Description

ISO C++ TR1 entities toplevel namespace is std::tr1.

4.24.2 Function Documentation

arg()

```
template<typename _Tp>
_Tp std::arg (
            const complex< _Tp > & __z)  [inline]
```

Return phase angle of *z*.

conf_hyperg()

```
template<typename _Tpa, typename _Tpc, typename _Tp>
__gnu_cxx::__promote_3< _Tpa, _Tpc, _Tp >::__type __gnu_cxx::conf_hyperg (
            _Tpa __a,
            _Tpc __c,
            _Tp __x)  [inline]
```

Return the confluent hypergeometric function ${}_1F_1(a;c;x)$ of real numeratorial parameter `a`, denominatorial parameter `c`, and argument `x`.
The confluent hypergeometric function is defined by

$$ {}_1F_1(a;c;x) = \sum_{n=0}^{\infty} \frac{(a)_n x^n}{(c)_n n!} $$

where the Pochhammer symbol is $(x)_k = (x)(x+1)...(x+k-1)$, $(x)_0 = 1$

Parameters

_↩ _a	The numeratorial parameter
_↩ _c	The denominatorial parameter
_↩ _x	The argument

conf_hypergf()

```
float __gnu_cxx::conf_hypergf (
             float __a,
             float __c,
             float __x)  [inline]
```

Return the confluent hypergeometric function ${}_1F_1(a;c;x)$ of `float` numeratorial parameter `a`, denominatorial parameter `c`, and argument `x`.

See also

conf_hyperg for details.

conf_hypergl()

```
long double __gnu_cxx::conf_hypergl (
             long double __a,
             long double __c,
             long double __x)  [inline]
```

Return the confluent hypergeometric function ${}_1F_1(a;c;x)$ of `long double` numeratorial parameter `a`, denominatorial parameter `c`, and argument `x`.

See also

conf_hyperg for details.

hyperg()

```
template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp>
__gnu_cxx::__promote_4< _Tpa, _Tpb, _Tpc, _Tp >::__type __gnu_cxx::hyperg (
             _Tpa __a,
             _Tpb __b,
             _Tpc __c,
             _Tp __x)  [inline]
```

Return the hypergeometric function ${}_2F_1(a,b;c;x)$ of real numeratorial parameters `a` and `b`, denominatorial parameter `c`, and argument `x`.
The hypergeometric function is defined by

$$ {}_2F_1(a;c;x) = \sum_{n=0}^{\infty} \frac{(a)_n (b)_n x^n}{(c)_n n!} $$

where the Pochhammer symbol is $(x)_k = (x)(x+1)...(x+k-1)$, $(x)_0 = 1$

Parameters

_↩ _a	The first numeratorial parameter
_↩ _b	The second numeratorial parameter
_↩ _c	The denominatorial parameter
_↩ _x	The argument

hypergf()

```
float __gnu_cxx::hypergf (
            float __a,
            float __b,
            float __c,
            float __x)  [inline]
```

Return the hypergeometric function ${}_2F_1(a,b;c;x)$ of @ float numeratorial parameters `a` and `b`, denominatorial parameter `c`, and argument `x`.

See also

hyperg for details.

hypergl()

```
long double __gnu_cxx::hypergl (
            long double __a,
            long double __b,
            long double __c,
            long double __x)  [inline]
```

Return the hypergeometric function ${}_2F_1(a,b;c;x)$ of `long double` numeratorial parameters `a` and `b`, denominatorial parameter `c`, and argument `x`.

See also

hyperg for details.

norm()

```
template<typename _Tp>
_Tp constexpr std::norm (
            const complex< _Tp > & )  [constexpr]
```

Return *z* magnitude squared.

polar()

```
template<typename _Tp>
complex< _Tp > std::polar (
            const _Tp & __rho,
            const _Tp & __theta = _Tp(0))  [inline]
```

Return complex with magnitude *rho* and angle *theta*.

4.25 std::tr1::__detail Namespace Reference

4.25.1 Detailed Description

Implementation details not part of the namespace std::tr1 interface.

4.26 std::tr2 Namespace Reference

Namespaces

- namespace __detail

Classes

- struct __dynamic_bitset_base
- struct __reflection_typelist
- struct __reflection_typelist< _First, _Rest... >
- struct __reflection_typelist<>
- struct bases
- class bool_set
- struct direct_bases
- class dynamic_bitset

Functions

- bool **certainly** (bool_set __b)
- bool **contains** (bool_set __s, bool_set __t)
- bool **equals** (bool_set __s, bool_set __t)
- bool **is_emptyset** (bool_set __b)
- bool **is_indeterminate** (bool_set __b)
- bool **is_singleton** (bool_set __b)
- bool_set **operator!=** (bool __s, bool_set __t)
- bool_set **operator!=** (bool_set __s, bool __t)
- bool_set **operator!=** (bool_set __s, bool_set __t)
- bool_set **operator&** (bool __s, bool_set __t)
- bool_set **operator&** (bool_set __s, bool __t)
- template<typename _CharT, typename _Traits, typename _WordT, typename _Alloc>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const dynamic_bitset< _WordT, _Alloc > &__x)
- bool_set **operator==** (bool __s, bool_set __t)
- bool_set **operator==** (bool_set __s, bool __t)
- template<typename _CharT, typename _Traits, typename _WordT, typename _Alloc>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, dynamic_bitset< _WordT, _Alloc > &__x)
- bool_set **operator^** (bool __s, bool_set __t)
- bool_set **operator^** (bool_set __s, bool __t)
- bool_set **operator|** (bool __s, bool_set __t)
- bool_set **operator|** (bool_set __s, bool __t)
- bool **possibly** (bool_set __b)
- bool_set **set_complement** (bool_set __b)
- bool_set **set_intersection** (bool __s, bool_set __t)
- bool_set **set_intersection** (bool_set __s, bool __t)
- bool_set **set_intersection** (bool_set __s, bool_set __t)

- bool_set **set_union** (bool __s, bool_set __t)
- bool_set **set_union** (bool_set __s, bool __t)
- bool_set **set_union** (bool_set __s, bool_set __t)

- template<typename _WordT, typename _Alloc>
  bool operator!= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)
- template<typename _WordT, typename _Alloc>
  bool operator<= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)
- template<typename _WordT, typename _Alloc>
  bool operator> (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)
- template<typename _WordT, typename _Alloc>
  bool operator>= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)

- template<typename _WordT, typename _Alloc>
  dynamic_bitset< _WordT, _Alloc > operator& (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)
- template<typename _WordT, typename _Alloc>
  dynamic_bitset< _WordT, _Alloc > operator| (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)
- template<typename _WordT, typename _Alloc>
  dynamic_bitset< _WordT, _Alloc > operator^ (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)
- template<typename _WordT, typename _Alloc>
  dynamic_bitset< _WordT, _Alloc > operator- (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)

4.26.1 Detailed Description

Namespace for non-standard "TR2" extensions.

4.27 std::tr2::__detail Namespace Reference

4.27.1 Detailed Description

Implementation details not part of the namespace std::tr2 interface.

5 Class Documentation

5.1 __gnu_parallel::__accumulate_binop_reduct< _BinOp > Struct Template Reference

`#include <for_each_selectors.h>`

Public Member Functions

- **__accumulate_binop_reduct** (_BinOp &__b)
- template<typename _Result, typename _Addend>
  _Result **operator()** (const _Result &__x, const _Addend &__y)

Public Attributes

- _BinOp & **__binop**

5.1.1 Detailed Description

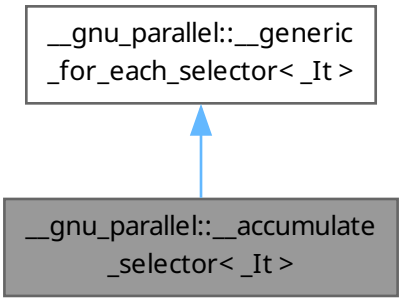
template<typename _BinOp>
struct __gnu_parallel::__accumulate_binop_reduct< _BinOp >

General reduction, using a binary operator.
The documentation for this struct was generated from the following file:

- for_each_selectors.h

5.2 __gnu_parallel::__accumulate_selector< _It > Struct Template Reference

`#include <for_each_selectors.h>`
Inheritance diagram for __gnu_parallel::__accumulate_selector< _It >:



Public Member Functions

- template<typename _Op>
  std::iterator_traits< _It >::value_type operator() (_Op __o, _It __i)

Public Attributes

- _It _M_finish_iterator

5.2.1 Detailed Description

template<typename _It>
struct __gnu_parallel::__accumulate_selector< _It >

std::accumulate() selector.

5.2.2 Member Function Documentation

operator()()

```
template<typename _It>
template<typename _Op>
std::iterator_traits< _It >::value_type __gnu_parallel::__accumulate_selector< _It >::operator()
(
            _Op __o,
            _It __i)  [inline]
```

Functor execution.

Parameters

_↩ _o	Operator (unused).
_↩ _i	iterator referencing object.

Returns

The current value.

5.2.3 Member Data Documentation

_M_finish_iterator

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator  [inherited]
```

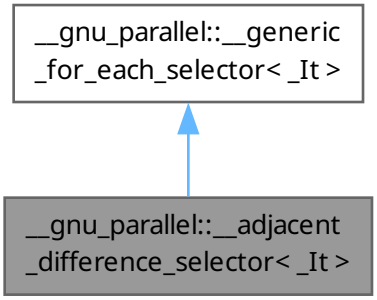
_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).
The documentation for this struct was generated from the following file:

- for_each_selectors.h

5.3 __gnu_parallel::__adjacent_difference_selector< _It > Struct Template Reference

```
#include <for_each_selectors.h>
```

Inheritance diagram for __gnu_parallel::__adjacent_difference_selector< _It >:



Public Member Functions

- template<typename _Op>
  bool **operator()** (_Op &__o, _It __i)

Public Attributes

- _It _M_finish_iterator

5.3.1 Detailed Description

template<typename _It>
struct __gnu_parallel::__adjacent_difference_selector< _It >

Selector that returns the difference between two adjacent __elements.

5.3.2 Member Data Documentation

_M_finish_iterator

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator  [inherited]
```

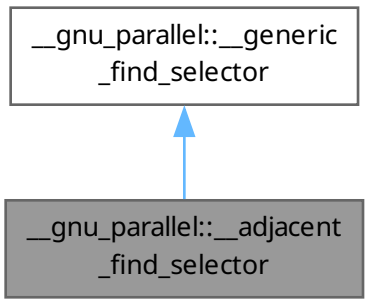
_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).
The documentation for this struct was generated from the following file:

- for_each_selectors.h

5.4 __gnu_parallel::__adjacent_find_selector Struct Reference

```
#include <find_selectors.h>
```

Inheritance diagram for __gnu_parallel::__adjacent_find_selector:



Public Member Functions

- template<typename _RAIter1, typename _RAIter2, typename _Pred>
  std::pair< _RAIter1, _RAIter2 > _M_sequential_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __↩
  begin2, _Pred __pred)
- template<typename _RAIter1, typename _RAIter2, typename _Pred>
  bool operator() (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)

5.4.1 Detailed Description

Test predicate on two adjacent elements.

5.4.2 Member Function Documentation

_M_sequential_algorithm()

```
template<typename _RAIter1, typename _RAIter2, typename _Pred>
std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__adjacent_find_selector::_M_sequential_algorithm
(
             _RAIter1 __begin1,
             _RAIter1 __end1,
             _RAIter2 __begin2,
             _Pred __pred)  [inline]
```

Corresponding sequential algorithm on a sequence.

Parameters

__begin1	Begin iterator of first sequence.
__end1	End iterator of first sequence.
__begin2	Begin iterator of second sequence.
__pred	Find predicate.

References std::make_pair().

operator()()

```
template<typename _RAIter1, typename _RAIter2, typename _Pred>
bool __gnu_parallel::__adjacent_find_selector::operator() (
             _RAIter1 __i1,
             _RAIter2 __i2,
             _Pred __pred)  [inline]
```

Test on one position.

Parameters

__i1	_Iterator on first sequence.
__i2	_Iterator on second sequence (unused).
__pred	Find predicate.

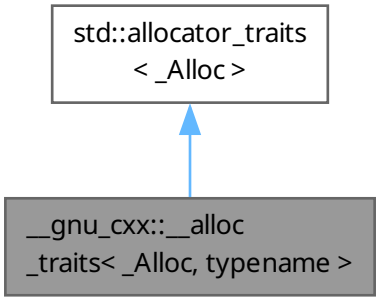
The documentation for this struct was generated from the following file:

- find_selectors.h

5.5 __gnu_cxx::__alloc_traits< _Alloc, typename > Struct Template Reference

```
#include <alloc_traits.h>
```

Inheritance diagram for __gnu_cxx::__alloc_traits< _Alloc, typename >:



Public Types

- typedef std::allocator_traits< _Alloc > **_Base_type**
- typedef _Alloc **allocator_type**
- typedef _Base_type::const_pointer **const_pointer**
- typedef const value_type & **const_reference**
- using const_void_pointer
- typedef _Base_type::difference_type **difference_type**
- using is_always_equal
- typedef _Base_type::pointer **pointer**
- using propagate_on_container_copy_assignment
- using propagate_on_container_move_assignment
- using propagate_on_container_swap
- template<typename _Tp>
  using **rebind_alloc**
- template<typename _Tp>
  using **rebind_traits**
- typedef value_type & **reference**
- typedef _Base_type::size_type **size_type**
- typedef _Base_type::value_type **value_type**
- using void_pointer

Static Public Member Functions

- static constexpr bool **_S_always_equal** ()
- static constexpr bool **_S_nothrow_move** ()
- static constexpr void **_S_on_swap** (_Alloc &__a, _Alloc &__b)
- static constexpr bool **_S_propagate_on_copy_assign** ()
- static constexpr bool **_S_propagate_on_move_assign** ()
- static constexpr bool **_S_propagate_on_swap** ()
- static constexpr _Alloc **_S_select_on_copy** (const _Alloc &__a)
- static constexpr pointer allocate (_Alloc &__a, size_type __n)
- static constexpr pointer allocate (_Alloc &__a, size_type __n, const_void_pointer __hint)

- static constexpr pointer allocate (_Alloc &__a, size_type __n)
- static constexpr pointer allocate (_Alloc &__a, size_type __n, const_void_pointer __hint)
- template<typename _Ptr, typename... _Args>
  static constexpr std::__enable_if_t< __is_custom_pointer< _Ptr >::value > **construct** (_Alloc &__a, _Ptr _↩
  _p, _Args &&... __args) noexcept(noexcept(_Base_type::construct(__a, std::__to_address(__p), std::forward<
  _Args >(__args)...)))
- template<typename _Tp, typename... _Args>
  requires __can_construct<_Alloc, _Tp, _Args...>
  static constexpr void construct (_Alloc &__a, _Tp *__p, _Args &&... __args) noexcept(_S_nothrow_construct<
  _Tp, _Args... >())
- static constexpr void deallocate (_Alloc &__a, pointer __p, size_type __n)
- static constexpr void deallocate (_Alloc &__a, pointer __p, size_type __n)
- template<typename _Ptr>
  static constexpr std::__enable_if_t< __is_custom_pointer< _Ptr >::value > **destroy** (_Alloc &__a, _Ptr __p)
  noexcept(noexcept(_Base_type::destroy(__a, std::__to_address(__p))))
- template<typename _Tp>
  static constexpr void destroy (_Alloc &__a, _Tp *__p) noexcept(_S_nothrow_destroy< _Tp >())
- static constexpr size_type max_size (const _Alloc &__a) noexcept
- static constexpr _Alloc select_on_container_copy_construction (const _Alloc &__rhs)

5.5.1 Detailed Description

template<typename _Alloc, typename = typename _Alloc::value_type>
struct __gnu_cxx::__alloc_traits< _Alloc, typename >

Uniform interface to C++98 and C++11 allocators.

5.5.2 Member Typedef Documentation

const_void_pointer

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::const_void_pointer  [inherited]
```

The allocator's const void pointer type.
`Alloc::const_void_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<const void>`

is_always_equal

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::is_always_equal  [inherited]
```

Whether all instances of the allocator type compare equal.
`Alloc::is_always_equal` if that type exists, otherwise `is_empty<Alloc>::type`

propagate_on_container_copy_assignment

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::propagate_on_container_copy_assignment  [inherited]
```

How the allocator is propagated on copy assignment.
`Alloc::propagate_on_container_copy_assignment` if that type exists, otherwise `false_type`

propagate_on_container_move_assignment

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::propagate_on_container_move_assignment  [inherited]
```

How the allocator is propagated on move assignment.
`Alloc::propagate_on_container_move_assignment` if that type exists, otherwise `false_type`

propagate_on_container_swap

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::propagate_on_container_swap  [inherited]
```

How the allocator is propagated on swap.
`Alloc::propagate_on_container_swap` if that type exists, otherwise `false_type`

void_pointer

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::void_pointer  [inherited]
```

The allocator's void pointer type.
`Alloc::void_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<void>`

5.5.3 Member Function Documentation

allocate() [1/4]

```
template<typename _Alloc, typename = typename _Alloc::value_type>
static constexpr pointer std::allocator_traits< _Alloc >::allocate (
            _Alloc & __a,
            size_type __n)  [inline], [static], [nodiscard], [constexpr]
```

Allocate memory.

Parameters

_a	An allocator.
_n	The number of objects to allocate space for.

Calls `a.allocate(n)`

allocate() [2/4]

```
template<typename _Alloc, typename = typename _Alloc::value_type>
static constexpr pointer std::allocator_traits< _Alloc >::allocate (
            _Alloc & __a,
            size_type __n,
            const_void_pointer __hint)  [inline], [static], [nodiscard], [constexpr]
```

Allocate memory.

Parameters

__a	An allocator.
__n	The number of objects to allocate space for.
__hint	Aid to locality.

Returns

Memory of suitable size and alignment for *n* objects of type `value_type`

Returns `a.allocate(n, hint)` if that expression is well-formed, otherwise returns `a.allocate(n)`

allocate() [3/4]

```
template<typename _Alloc>
static constexpr pointer std::allocator_traits< _Alloc >::allocate (
            _Alloc & __a,
            size_type __n)  [inline], [static], [nodiscard], [constexpr], [inherited]
```

Allocate memory.

Parameters

_↩ _a	An allocator.
_↩ _n	The number of objects to allocate space for.

Calls `a.allocate(n)`

allocate() [4/4]

```
template<typename _Alloc>
static constexpr pointer std::allocator_traits< _Alloc >::allocate (
            _Alloc & __a,
            size_type __n,
            const_void_pointer __hint)  [inline], [static], [nodiscard], [constexpr], [inherited]
```

Allocate memory.

Parameters

__a	An allocator.
__n	The number of objects to allocate space for.
__hint	Aid to locality.

Returns

Memory of suitable size and alignment for *n* objects of type `value_type`

Returns `a.allocate(n, hint)` if that expression is well-formed, otherwise returns `a.allocate(n)`

construct()

```
template<typename _Alloc, typename = typename _Alloc::value_type>
template<typename _Tp, typename...  _Args>
requires __can_construct<_Alloc, _Tp, _Args...>
static constexpr void std::allocator_traits< _Alloc >::construct (
            _Alloc & __a,
            _Tp * __p,
            _Args &&...  __args)  [inline], [static], [constexpr], [noexcept]
```

Construct an object of type `_Tp`

Parameters

__a	An allocator.
__p	Pointer to memory of suitable size and alignment for Tp
__args	Constructor arguments.

Calls `__a.construct(__p, std::forward<Args>(__args)...)` if that expression is well-formed, otherwise uses placement-new to construct an object of type *_Tp* at location *__p* from the arguments *__args*...

deallocate() [1/2]

```
template<typename _Alloc, typename = typename _Alloc::value_type>
static constexpr void std::allocator_traits< _Alloc >::deallocate (
            _Alloc & __a,
            pointer __p,
            size_type __n)  [inline], [static], [constexpr]
```

Deallocate memory.

Parameters

_a	An allocator.
_p	Pointer to the memory to deallocate.
_n	The number of objects space was allocated for.

Calls `a.deallocate(p, n)`

deallocate() [2/2]

```
template<typename _Alloc>
static constexpr void std::allocator_traits< _Alloc >::deallocate (
            _Alloc & __a,
            pointer __p,
            size_type __n)  [inline], [static], [constexpr], [inherited]
```

Deallocate memory.

Parameters

_a	An allocator.
_p	Pointer to the memory to deallocate.
_n	The number of objects space was allocated for.

Calls `a.deallocate(p, n)`

destroy()

```
template<typename _Alloc, typename = typename _Alloc::value_type>
template<typename _Tp>
static constexpr void std::allocator_traits< _Alloc >::destroy (
            _Alloc & __a,
            _Tp * __p)  [inline], [static], [constexpr], [noexcept]
```

Destroy an object of type *_Tp*.

Parameters

_↩ _a	An allocator.
_↩ _p	Pointer to the object to destroy

Calls `__a.destroy(__p)` if that expression is well-formed, otherwise calls `__p->~_Tp()`

max_size()

```
template<typename _Alloc, typename = typename _Alloc::value_type>
static constexpr size_type std::allocator_traits< _Alloc >::max_size (
            const _Alloc & __a)  [inline], [static], [constexpr], [noexcept]
```

The maximum supported allocation size.

Parameters

_↩ _a	An allocator.

Returns

`__a.max_size()` or `numeric_limits<size_type>::max()`

Returns `__a.max_size()` if that expression is well-formed, otherwise returns `numeric_limits<size_↩ type>::max()`

select_on_container_copy_construction()

```
template<typename _Alloc>
static constexpr _Alloc std::allocator_traits< _Alloc >::select_on_container_copy_construction (
            const _Alloc & __rhs)  [inline], [static], [constexpr], [inherited]
```

Obtain an allocator to use when copying a container.

Parameters

__rhs	An allocator.

Returns

`__rhs.select_on_container_copy_construction()` or *__rhs*

Returns `__rhs.select_on_container_copy_construction()` if that expression is well-formed, otherwise returns *__rhs*

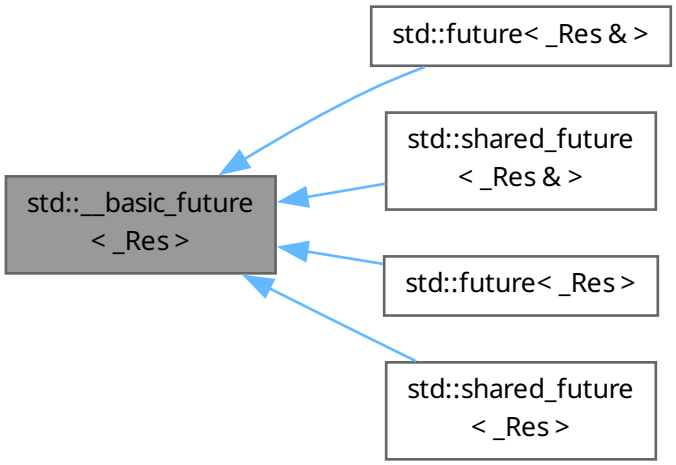
The documentation for this struct was generated from the following file:

- ext/alloc_traits.h

5.6 std::__basic_future< _Res > Class Template Reference

`#include <future>`

Inheritance diagram for std::__basic_future< _Res >:



Public Member Functions

- **__basic_future** (const __basic_future &)=delete
- __basic_future & **operator=** (const __basic_future &)=delete
- bool **valid** () const noexcept
- void **wait** () const
- template<typename _Rep, typename _Period>
  future_status **wait_for** (const chrono::duration< _Rep, _Period > &__rel) const
- template<typename _Clock, typename _Duration>
  future_status **wait_until** (const chrono::time_point< _Clock, _Duration > &__abs) const

Protected Types

- typedef __future_base::_Result< _Res > & **__result_type**
- typedef shared_ptr< _State_base > **__state_type**

Protected Member Functions

- **__basic_future** (const __state_type &__state)
- **__basic_future** (const shared_future< _Res > &) noexcept
- **__basic_future** (future< _Res > &&) noexcept
- **__basic_future** (shared_future< _Res > &&) noexcept
- __result_type _M_get_result () const
- void **_M_swap** (__basic_future &__that) noexcept

5.6.1 Detailed Description

template<typename _Res>
class std::__basic_future< _Res >

Common implementation for future and shared_future.

5.6.2 Member Function Documentation

_M_get_result()

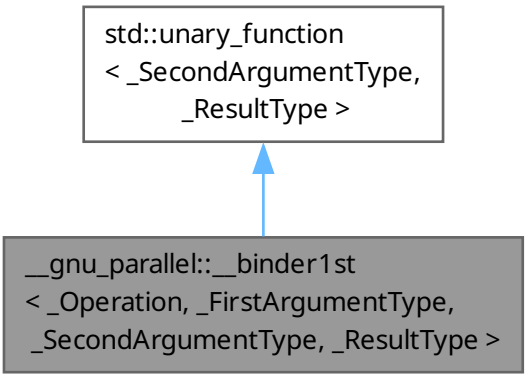
```
template<typename _Res>
__result_type std::__basic_future< _Res >::_M_get_result () const  [inline], [protected]
```

Wait for the state to be ready and rethrow any stored exception.
The documentation for this class was generated from the following file:

- future

5.7 __gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType > Class Template Reference

`#include <base.h>`
Inheritance diagram for __gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >:



Public Types

- typedef _SecondArgumentType argument_type
- typedef _ResultType result_type

Public Member Functions

- **__binder1st** (const _Operation &__x, const _FirstArgumentType &__y)
- _ResultType **operator()** (_SecondArgumentType &__x) const
- _ResultType **operator()** (const _SecondArgumentType &__x)

Protected Attributes

- _Operation **_M_op**
- _FirstArgumentType **_M_value**

5.7.1 Detailed Description

template<typename _Operation, typename _FirstArgumentType, typename _SecondArgumentType, typename _ResultType>
class __gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >

Similar to std::binder1st, but giving the argument types explicitly.

5.7.2 Member Typedef Documentation

argument_type

```
typedef _SecondArgumentType std::unary_function< _SecondArgumentType, _ResultType >::argument_↩
type [inherited]
```

`argument_type` is the type of the argument

result_type

```
typedef _ResultType std::unary_function< _SecondArgumentType, _ResultType >::result_type [inherited]
```

`result_type` is the return type

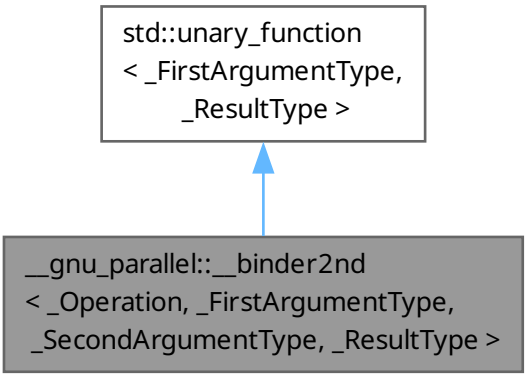
The documentation for this class was generated from the following file:

- base.h

5.8 __gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType > Class Template Reference

```
#include <base.h>
```

Inheritance diagram for __gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _↩ ResultType >:



Public Types

- typedef _FirstArgumentType argument_type
- typedef _ResultType result_type

Public Member Functions

- **__binder2nd** (const _Operation &__x, const _SecondArgumentType &__y)
- _ResultType **operator()** (_FirstArgumentType &__x)
- _ResultType **operator()** (const _FirstArgumentType &__x) const

Protected Attributes

- _Operation **_M_op**
- _SecondArgumentType **_M_value**

5.8.1 Detailed Description

template<typename _Operation, typename _FirstArgumentType, typename _SecondArgumentType, typename _ResultType>
class __gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >

Similar to std::binder2nd, but giving the argument types explicitly.

5.8.2 Member Typedef Documentation

argument_type

`typedef _FirstArgumentType std::unary_function< _FirstArgumentType, _ResultType >::argument_type [inherited]`
`argument_type` is the type of the argument

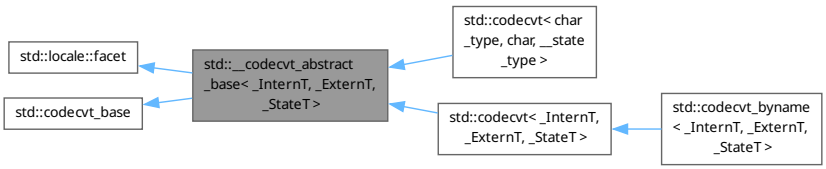
result_type

`typedef _ResultType std::unary_function< _FirstArgumentType, _ResultType >::result_type [inherited]`
`result_type` is the return type
The documentation for this class was generated from the following file:

- base.h

5.9 std::__codecvt_abstract_base< _InternT, _ExternT, _StateT > Class Template Reference

`#include <codecvt.h>`
Inheritance diagram for std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >:



Public Types

- typedef _ExternT **extern_type**
- typedef _InternT **intern_type**
- typedef codecvt_base::result **result**
- typedef _StateT **state_type**

Public Member Functions

- bool **always_noconv** () const throw ()
- int **encoding** () const throw ()
- result in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- int **length** (state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max) const
- int **max_length** () const throw ()
- result out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const

Protected Member Functions

- **__codecvt_abstract_base** (size_t __refs=0)
- virtual bool **do_always_noconv** () const throw () =0
- virtual int **do_encoding** () const throw () =0
- virtual result **do_in** (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const =0
- virtual int **do_length** (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const =0
- virtual int **do_max_length** () const throw () =0
- virtual result do_out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const =0
- virtual result **do_unshift** (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to↩ _next) const =0

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

5.9.1 Detailed Description

template<typename _InternT, typename _ExternT, typename _StateT>
class std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >

Common base for codecvt functions.
This template class provides implementations of the public functions that forward to the protected virtual functions.
This template also provides abstract stubs for the protected virtual functions.

5.9.2 Member Function Documentation

do_out()

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::do_out (
            state_type & __state,
            const intern_type * __from,
            const intern_type * __from_end,
            const intern_type *& __from_next,
```

```
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [protected], [pure virtual]
```

Convert from internal to external character set.
Converts input string of intern_type to output string of extern_type. This function is a hook for derived classes to change the value returned.

See also

out for more information.

Implemented in std::codecvt< _InternT, _ExternT, _StateT >, std::codecvt< _Elem, char, mbstate_t >, std::codecvt< _InternT, _ExternT
std::codecvt< _InternT, _ExternT, encoding_state >, std::codecvt< char, char, mbstate_t >, std::codecvt< char, char, mbstate_t >,
std::codecvt< char16_t, char, mbstate_t >, std::codecvt< char16_t, char, mbstate_t >, std::codecvt< char32_t, char, mbstate_t >,
std::codecvt< char32_t, char, mbstate_t >, std::codecvt< wchar_t, char, mbstate_t >, and std::codecvt< wchar_t, char, mbstate_t >.
Referenced by out().

in()

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::in (
            state_type & __state,
            const extern_type * __from,
            const extern_type * __from_end,
            const extern_type *& __from_next,
            intern_type * __to,
            intern_type * __to_end,
            intern_type *& __to_next) const  [inline]
```

Convert from external to internal character set.
Converts input string of extern_type to output string of intern_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do_in.
The source and destination character sets are determined by the facet's locale, internal and external types.
The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.
The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.
The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

out()

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::out (
             state_type & __state,
             const intern_type * __from,
             const intern_type * __from_end,
             const intern_type *& __from_next,
             extern_type * __to,
             extern_type * __to_end,
             extern_type *& __to_next) const  [inline]
```

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This is analogous to wcsrtombs. It does this by calling codecvt::do_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

References do_out().

unshift()

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::unshift (
             state_type & __state,
             extern_type * __to,
             extern_type * __to_end,
             extern_type *& __to_next) const  [inline]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.
The result returned is a member of codecvt_base::result. If the state could be reset and data written, returns codecvt↩ _base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the output has insufficient space, returns codecvt_base::partial. Otherwise the reset failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

The documentation for this class was generated from the following file:

- codecvt.h

5.10 __gnu_cxx::__common_pool_policy< _PoolTp, _Thread > Struct Template Reference

`#include <mt_allocator.h>`

5.10.1 Detailed Description

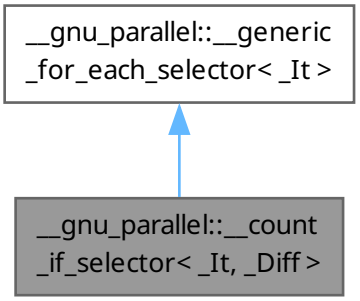
template<template< bool > class _PoolTp, bool _Thread>
struct __gnu_cxx::__common_pool_policy< _PoolTp, _Thread >

Policy for shared __pool objects.
The documentation for this struct was generated from the following file:

- mt_allocator.h

5.11 __gnu_parallel::__count_if_selector< _It, _Diff > Struct Template Reference

`#include <for_each_selectors.h>`
Inheritance diagram for __gnu_parallel::__count_if_selector< _It, _Diff >:



Public Member Functions

- template<typename _Op>
  _Diff operator() (_Op &__o, _It __i)

Public Attributes

- _It _M_finish_iterator

5.11.1 Detailed Description

template<typename _It, typename _Diff>
struct __gnu_parallel::__count_if_selector< _It, _Diff >

std::count_if () selector.

5.11.2 Member Function Documentation

operator()()

```
template<typename _It, typename _Diff>
template<typename _Op>
_Diff __gnu_parallel::__count_if_selector< _It, _Diff >::operator() (
            _Op & __o,
            _It __i)  [inline]
```

Functor execution.

Parameters

_↩ _o	Operator.
_↩ _i	iterator referencing object.

Returns

1 if count, 0 if does not count.

5.11.3 Member Data Documentation

_M_finish_iterator

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator  [inherited]
```

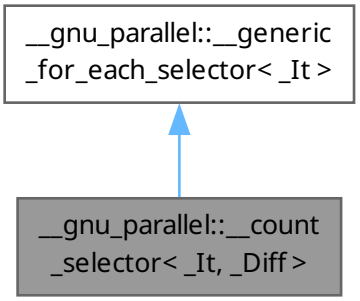
_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).
The documentation for this struct was generated from the following file:

- for_each_selectors.h

5.12 __gnu_parallel::__count_selector< _It, _Diff > Struct Template Reference

```
#include <for_each_selectors.h>
```

Inheritance diagram for __gnu_parallel::__count_selector< _It, _Diff >:



Public Member Functions

- template<typename _ValueType>
  _Diff operator() (_ValueType &__v, _It __i)

Public Attributes

- _It _M_finish_iterator

5.12.1 Detailed Description

template<typename _It, typename _Diff>
struct __gnu_parallel::__count_selector< _It, _Diff >

std::count() selector.

5.12.2 Member Function Documentation

operator()()

```
template<typename _It, typename _Diff>
template<typename _ValueType>
_Diff __gnu_parallel::__count_selector< _It, _Diff >::operator() (
            _ValueType & __v,
            _It __i)  [inline]
```

Functor execution.

Parameters

_↩ _v	Current value.
_↩ _i	iterator referencing object.

Returns

1 if count, 0 if does not count.

5.12.3 Member Data Documentation

_M_finish_iterator

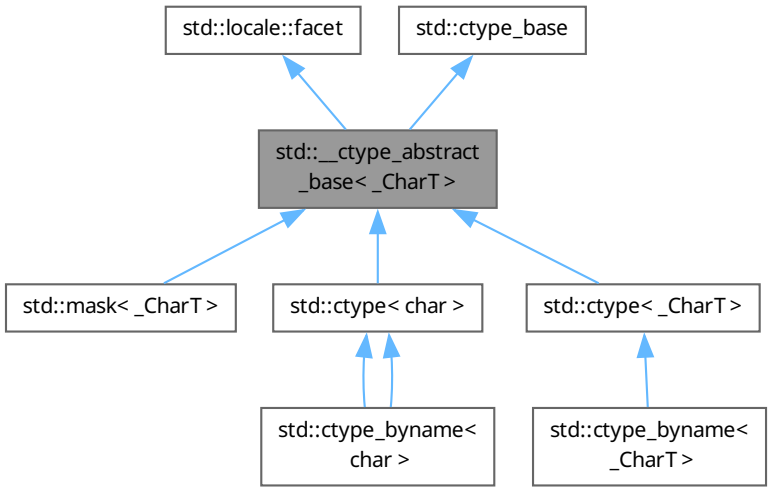
```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator  [inherited]
```

_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).
The documentation for this struct was generated from the following file:

- for_each_selectors.h

5.13 std::__ctype_abstract_base< _CharT > Class Template Reference

`#include <locale_facets.h>`
Inheritance diagram for std::__ctype_abstract_base< _CharT >:



Public Types

- typedef const int * **__to_type**
- typedef _CharT char_type
- typedef unsigned short **mask**

Public Member Functions

- const char_type * is (const char_type *__lo, const char_type *__hi, mask *__vec) const
- bool is (mask __m, char_type __c) const
- char narrow (char_type __c, char __dfault) const
- const char_type * narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__to) const
- const char_type * scan_is (mask __m, const char_type *__lo, const char_type *__hi) const

- const char_type * scan_not (mask __m, const char_type *__lo, const char_type *__hi) const
- const char_type * tolower (char_type *__lo, const char_type *__hi) const
- char_type tolower (char_type __c) const
- const char_type * toupper (char_type *__lo, const char_type *__hi) const
- char_type toupper (char_type __c) const
- char_type widen (char __c) const
- const char * widen (const char *__lo, const char *__hi, char_type *__to) const

Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **blank**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

Protected Member Functions

- **__ctype_abstract_base** (size_t __refs=0)
- virtual const char_type * do_is (const char_type *__lo, const char_type *__hi, mask *__vec) const =0
- virtual bool do_is (mask __m, char_type __c) const =0
- virtual char do_narrow (char_type __c, char __dfault) const =0
- virtual const char_type * do_narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__to) const =0
- virtual const char_type * do_scan_is (mask __m, const char_type *__lo, const char_type *__hi) const =0
- virtual const char_type * do_scan_not (mask __m, const char_type *__lo, const char_type *__hi) const =0
- virtual const char_type * do_tolower (char_type *__lo, const char_type *__hi) const =0
- virtual char_type do_tolower (char_type __c) const =0
- virtual const char_type * do_toupper (char_type *__lo, const char_type *__hi) const =0
- virtual char_type do_toupper (char_type __c) const =0
- virtual char_type do_widen (char __c) const =0
- virtual const char * do_widen (const char *__lo, const char *__hi, char_type *__to) const =0

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

5.13.1 Detailed Description

template<typename _CharT>
class std::__ctype_abstract_base< _CharT >

Common base for ctype facet.
This template class provides implementations of the public functions that forward to the protected virtual functions.
This template also provides abstract stubs for the protected virtual functions.

5.13.2 Member Typedef Documentation

char_type

```
template<typename _CharT>
typedef _CharT std::__ctype_abstract_base< _CharT >::char_type
```

Typedef for the template parameter.

5.13.3 Member Function Documentation

do_is() [1/2]

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base< _CharT >::do_is (
            const char_type * __lo,
            const char_type * __hi,
            mask * __vec) const  [protected], [pure virtual]
```

Return a mask array.
This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.
do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__vec	Pointer to an array of mask storage.

Returns

__hi.

Implemented in std::ctype< _CharT >, std::ctype< wchar_t >, std::ctype< wchar_t >, std::mask< _CharT >, and std::mask< _CharT >.

do_is() [2/2]

```
template<typename _CharT>
virtual bool std::__ctype_abstract_base< _CharT >::do_is (
            mask __m,
            char_type __c) const  [protected], [pure virtual]
```

Test char_type classification.
This function finds a mask M for *c* and compares it to mask *m*.
do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

_↩ _c	The char_type to find the mask of.
_↩ _m	The mask to compare against.

Returns

(M & __m) != 0.

Implemented in std::ctype< _CharT >, std::ctype< wchar_t >, std::ctype< wchar_t >, std::mask< _CharT >, and std::mask< _CharT >.
Referenced by std::mask< _CharT >::is(), and std::mask< _CharT >::is().

do_narrow() [1/2]

```
template<typename _CharT>
virtual char std::__ctype_abstract_base< _CharT >::do_narrow (
            char_type __c,
            char __dfault) const  [protected], [pure virtual]
```

Narrow char_type to char.
This virtual function converts the argument to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead.
do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char_type to convert.
__dfault	Char to return if conversion fails.

Returns

The converted char.

Implemented in std::ctype< _CharT >, std::ctype< wchar_t >, std::ctype< wchar_t >, std::mask< _CharT >, and std::mask< _CharT >.
Referenced by std::mask< _CharT >::narrow(), and std::mask< _CharT >::narrow().

do_narrow() [2/2]

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base< _CharT >::do_narrow (
            const char_type * __lo,
            const char_type * __hi,
            char __dfault,
            char * __to) const  [protected], [pure virtual]
```

Narrow char_type array to char.
This virtual function converts each char_type in the range [__lo,__hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, *__dfault* is used instead.
do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

Implemented in std::ctype< _CharT >, std::ctype< wchar_t >, std::ctype< wchar_t >, std::mask< _CharT >, and std::mask< _CharT >.

do_scan_is()

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base< _CharT >::do_scan_is (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [protected], [pure virtual]
```

Find char_type matching mask.
This function searches for and returns the first char_type c in [__lo,__hi) for which is(__m,c) is true.
do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

__m	The mask to compare against.
__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

Pointer to a matching char_type if found, else *__hi*.

Implemented in std::ctype< _CharT >, std::ctype< wchar_t >, std::ctype< wchar_t >, std::mask< _CharT >, and std::mask< _CharT >.
Referenced by std::mask< _CharT >::scan_is().

do_scan_not()

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base< _CharT >::do_scan_not (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [protected], [pure virtual]
```

Find char_type not matching mask.
This function searches for and returns a pointer to the first char_type c of [lo,hi) for which is(m,c) is false.
do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to a non-matching char_type if found, else __hi.

Implemented in std::ctype< _CharT >, std::ctype< wchar_t >, std::ctype< wchar_t >, std::mask< _CharT >, and std::mask< _CharT >.
Referenced by std::mask< _CharT >::scan_not().

do_tolower() [1/2]

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base< _CharT >::do_tolower (
            char_type * __lo,
            const char_type * __hi) const  [protected], [pure virtual]
```

Convert array to lowercase.
This virtual function converts each char_type in the range [__lo,__hi) to lowercase if possible. Other elements remain untouched.
do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

Implemented in std::ctype< _CharT >, std::ctype< wchar_t >, std::ctype< wchar_t >, std::mask< _CharT >, and std::mask< _CharT >.

do_tolower() [2/2]

```
template<typename _CharT>
virtual char_type std::__ctype_abstract_base< _CharT >::do_tolower (
            char_type __c) const  [protected], [pure virtual]
```

Convert to lowercase.
This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.
do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

_↩ _c	The char_type to convert.

Returns

The lowercase char_type if convertible, else __c.

Implemented in std::ctype< _CharT >, std::ctype< wchar_t >, std::ctype< wchar_t >, std::mask< _CharT >, and std::mask< _CharT >.
Referenced by std::mask< _CharT >::tolower(), and std::mask< _CharT >::tolower().

do_toupper() [1/2]

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base< _CharT >::do_toupper (
            char_type * __lo,
            const char_type * __hi) const  [protected], [pure virtual]
```

Convert array to uppercase.
This virtual function converts each char_type in the range [__lo,__hi) to uppercase if possible. Other elements remain untouched.
do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

Implemented in std::ctype< _CharT >, std::ctype< wchar_t >, std::ctype< wchar_t >, std::mask< _CharT >, and std::mask< _CharT >.

do_toupper() [2/2]

```
template<typename _CharT>
virtual char_type std::__ctype_abstract_base< _CharT >::do_toupper (
            char_type __c) const  [protected], [pure virtual]
```

Convert to uppercase.
This virtual function converts the char_type argument to uppercase if possible. If not possible (for example, '2'), returns the argument.
do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

_↩ _c	The char_type to convert.

Returns

The uppercase char_type if convertible, else __c.

Implemented in std::ctype< _CharT >, std::ctype< wchar_t >, std::ctype< wchar_t >, std::mask< _CharT >, and std::mask< _CharT >.
Referenced by std::mask< _CharT >::toupper(), and std::mask< _CharT >::toupper().

do_widen() [1/2]

```
template<typename _CharT>
virtual char_type std::__ctype_abstract_base< _CharT >::do_widen (
            char __c) const  [protected], [pure virtual]
```

Widen char.
This virtual function converts the char to char_type using the simplest reasonable transformation.
do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _c	The char to convert.

Returns

The converted char_type

Implemented in std::ctype< _CharT >, std::ctype< char >, std::ctype< char >, std::ctype< wchar_t >, std::ctype< wchar_t >, std::mask< _CharT >, and std::mask< _CharT >.
Referenced by std::mask< _CharT >::widen(), and std::mask< _CharT >::widen().

do_widen() [2/2]

```
template<typename _CharT>
virtual const char * std::__ctype_abstract_base< _CharT >::do_widen (
            const char * __lo,
            const char * __hi,
            char_type * __to) const  [protected], [pure virtual]
```

Widen char array.
This function converts each char in the input to char_type using the simplest reasonable transformation.
do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _lo	Pointer to start range.
_↩ _hi	Pointer to end of range.
_↩ _to	Pointer to the destination array.

Returns

__hi.

Implemented in std::ctype< _CharT >, std::ctype< wchar_t >, std::ctype< wchar_t >, std::mask< _CharT >, and std::mask< _CharT >.

is() [1/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::is (
            const char_type * __lo,
            const char_type * __hi,
            mask * __vec) const  [inline]
```

Return a mask array.
This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char_type>::do_is().

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__vec	Pointer to an array of mask storage.

Returns

__hi.

is() [2/2]

```
template<typename _CharT>
bool std::__ctype_abstract_base< _CharT >::is (
            mask __m,
            char_type __c) const  [inline]
```

Test char_type classification.
This function finds a mask M for *__c* and compares it to mask *__m*. It does so by returning the value of ctype<char_↩ type>::do_is().

Parameters

_↩ _*c*	The char_type to compare the mask of.
_↩ _*m*	The mask to compare against.

Returns

(M & __m) != 0.

Referenced by std::time_get< _CharT, _InIter >::get().

narrow() [1/2]

```
template<typename _CharT>
char std::__ctype_abstract_base< _CharT >::narrow (
            char_type __c,
            char __dfault) const  [inline]
```

Narrow char_type to char.
This function converts the char_type to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. It does so by returning ctype<char_type>::do_narrow(__c).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char_type to convert.
__dfault	Char to return if conversion fails.

Returns

The converted char.

Referenced by std::time_get< _CharT, _InIter >::do_get_year(), std::time_get< _CharT, _InIter >::get(), and std::time_put< _CharT, _OutIter >::put().

narrow() [2/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::narrow (
            const char_type * __lo,
            const char_type * __hi,
            char __dfault,
            char * __to) const  [inline]
```

Narrow array to char array.
This function converts each char_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char_type in the input that cannot be converted, *dfault* is used instead. It does so by returning ctype<char_type>::do_narrow(__lo, __hi, __dfault, __to).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

scan_is()

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::scan_is (
            mask __m,
```

```
            const char_type * __lo,
            const char_type * __hi) const  [inline]
```

Find char_type matching a mask.
This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char_type>::do_scan_is().

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to matching char_type if found, else __hi.

scan_not()

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::scan_not (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [inline]
```

Find char_type not matching a mask.
This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char_type>::do_scan_not().

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to first char in range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to non-matching char if found, else __hi.

tolower() [1/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::tolower (
            char_type * __lo,
            const char_type * __hi) const  [inline]
```

Convert array to lowercase.
This function converts each char_type in the range [__lo,__hi) to lowercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>:: do_tolower(__lo, __hi).

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

tolower() [2/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::tolower (
            char_type __c) const  [inline]
```

Convert to lowercase.
This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_tolower(c).

Parameters

_↩ _c	The char_type to convert.

Returns

The lowercase char_type if convertible, else __c.

Referenced by std::time_get< _CharT, _InIter >::get().

toupper() [1/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::toupper (
            char_type * __lo,
            const char_type * __hi) const  [inline]
```

Convert array to uppercase.
This function converts each char_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>:: do_toupper(lo, hi).

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

toupper() [2/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::toupper (
            char_type __c) const  [inline]
```

Convert to uppercase.
This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_toupper().

Parameters

_↩ _c	The char_type to convert.

Returns

The uppercase char_type if convertible, else __c.

Referenced by std::time_get< _CharT, _InIter >::get().

widen() [1/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::widen (
            char __c) const  [inline]
```

Widen char to char_type.
This function converts the char argument to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _c	The char to convert.

Returns

The converted char_type.

Referenced by std::money_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::money_put< _CharT, _Out and std::time_put< _CharT, _OutIter >::do_put().

widen() [2/2]

```
template<typename _CharT>
const char * std::__ctype_abstract_base< _CharT >::widen (
            const char * __lo,
            const char * __hi,
            char_type * __to) const  [inline]
```

Widen array to char_type.
This function converts each char in the input to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__to	Pointer to the destination array.

Returns

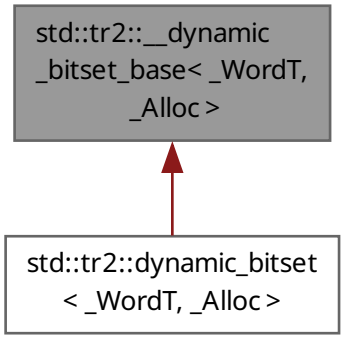
__hi.

The documentation for this class was generated from the following file:

- locale_facets.h

5.14 std::tr2::__dynamic_bitset_base< _WordT, _Alloc > Struct Template Reference

`#include <dynamic_bitset>`
Inheritance diagram for std::tr2::__dynamic_bitset_base< _WordT, _Alloc >:



Public Types

- typedef _Alloc **allocator_type**
- typedef _WordT **block_type**
- typedef size_t **size_type**

Public Member Functions

- **__dynamic_bitset_base** (__dynamic_bitset_base &&__b)=default
- **__dynamic_bitset_base** (const __dynamic_bitset_base &)=default
- **__dynamic_bitset_base** (const allocator_type &__alloc)
- **__dynamic_bitset_base** (size_type __nbits, unsigned long long __val=0ULL, const allocator_type &__alloc=allocator_type())

- size_t **_M_are_all_aux** () const noexcept
- void **_M_clear** () noexcept
- void **_M_do_and** (const __dynamic_bitset_base &__x) noexcept
- void **_M_do_append_block** (block_type __block, size_type __pos)
- size_t **_M_do_count** () const noexcept
- void **_M_do_dif** (const __dynamic_bitset_base &__x) noexcept
- size_type **_M_do_find_first** (size_t __not_found) const
- size_type **_M_do_find_next** (size_t __prev, size_t __not_found) const
- void **_M_do_flip** () noexcept
- void **_M_do_left_shift** (size_t __shift)
- void **_M_do_or** (const __dynamic_bitset_base &__x) noexcept
- void **_M_do_reset** () noexcept
- void **_M_do_right_shift** (size_t __shift)
- void **_M_do_set** () noexcept
- unsigned long long **_M_do_to_ullong** () const
- unsigned long **_M_do_to_ulong** () const
- void **_M_do_xor** (const __dynamic_bitset_base &__x) noexcept
- allocator_type **_M_get_allocator** () const noexcept
- block_type **_M_getword** (size_type __pos) const noexcept
- block_type & **_M_getword** (size_type __pos) noexcept
- block_type **_M_hiword** () const noexcept
- block_type & **_M_hiword** () noexcept
- bool **_M_is_any** () const noexcept
- bool **_M_is_equal** (const __dynamic_bitset_base &__x) const noexcept
- bool **_M_is_less** (const __dynamic_bitset_base &__x) const noexcept
- bool **_M_is_proper_subset_of** (const __dynamic_bitset_base &__b) const noexcept
- bool **_M_is_subset_of** (const __dynamic_bitset_base &__b) noexcept
- void **_M_resize** (size_t __nbits, bool __value)
- size_type **_M_size** () const noexcept
- void **_M_swap** (__dynamic_bitset_base &__b) noexcept
- __dynamic_bitset_base & **operator=** (__dynamic_bitset_base &&)=default
- __dynamic_bitset_base & **operator=** (const __dynamic_bitset_base &)=default

Static Public Member Functions

- static block_type **_S_maskbit** (size_type __pos) noexcept
- static size_type **_S_whichbit** (size_type __pos) noexcept
- static size_type **_S_whichbyte** (size_type __pos) noexcept
- static size_type **_S_whichword** (size_type __pos) noexcept

Public Attributes

- std::vector< block_type, allocator_type > _M_w

Static Public Attributes

- static const size_type **_S_bits_per_block**
- static const size_type **npos**

5.14.1 Detailed Description

template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
struct std::tr2::__dynamic_bitset_base< _WordT, _Alloc >

Base class, general case.
See documentation for dynamic_bitset.

5.14.2 Member Data Documentation

_M_w

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::vector<block_type, allocator_type> std::tr2::__dynamic_bitset_base< _WordT, _Alloc >::_M_w
```

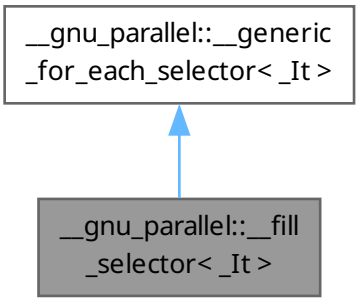
0 is the least significant word.
The documentation for this struct was generated from the following files:

- dynamic_bitset
- dynamic_bitset.tcc

5.15 __gnu_parallel::__fill_selector< _It > Struct Template Reference

```
#include <for_each_selectors.h>
```

Inheritance diagram for __gnu_parallel::__fill_selector< _It >:



Public Member Functions

- template<typename _ValueType>
  bool operator() (_ValueType &__v, _It __i)

Public Attributes

- _It _M_finish_iterator

5.15.1 Detailed Description

template<typename _It>
struct __gnu_parallel::__fill_selector< _It >

std::fill() selector.

5.15.2 Member Function Documentation

operator()()

```
template<typename _It>
template<typename _ValueType>
bool __gnu_parallel::__fill_selector< _It >::operator() (
            _ValueType & __v,
            _It __i)  [inline]
```

Functor execution.

Parameters

_↩ _v	Current value.
_↩ _i	iterator referencing object.

5.15.3 Member Data Documentation

_M_finish_iterator

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator  [inherited]
```

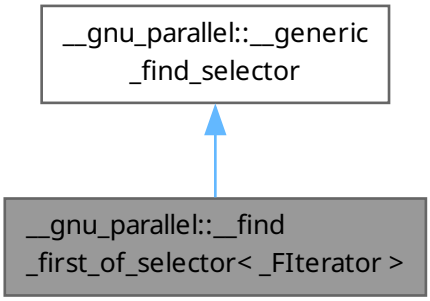
_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).
The documentation for this struct was generated from the following file:

- for_each_selectors.h

5.16 __gnu_parallel::__find_first_of_selector< _FIterator > Struct Template Reference

```
#include <find_selectors.h>
```

Inheritance diagram for __gnu_parallel::__find_first_of_selector< _FIterator >:



Public Member Functions

- **__find_first_of_selector** (_FIterator __begin, _FIterator __end)

- template<typename _RAIter1, typename _RAIter2, typename _Pred>
  std::pair< _RAIter1, _RAIter2 > _M_sequential_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __↩
  begin2, _Pred __pred)
- template<typename _RAIter1, typename _RAIter2, typename _Pred>
  bool operator() (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)

Public Attributes

- _FIterator **_M_begin**
- _FIterator **_M_end**

5.16.1 Detailed Description

template<typename _FIterator>
struct __gnu_parallel::__find_first_of_selector< _FIterator >

Test predicate on several elements.

5.16.2 Member Function Documentation

_M_sequential_algorithm()

```
template<typename _FIterator>
template<typename _RAIter1, typename _RAIter2, typename _Pred>
std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_first_of_selector< _FIterator >::_M_↩
sequential_algorithm (
            _RAIter1 __begin1,
            _RAIter1 __end1,
            _RAIter2 __begin2,
            _Pred __pred)  [inline]
```

Corresponding sequential algorithm on a sequence.

Parameters

__begin1	Begin iterator of first sequence.
__end1	End iterator of first sequence.
__begin2	Begin iterator of second sequence.
__pred	Find predicate.

References std::make_pair().

operator()()

```
template<typename _FIterator>
template<typename _RAIter1, typename _RAIter2, typename _Pred>
bool __gnu_parallel::__find_first_of_selector< _FIterator >::operator() (
            _RAIter1 __i1,
            _RAIter2 __i2,
            _Pred __pred)  [inline]
```

Test on one position.

Parameters

__i1	_Iterator on first sequence.
__i2	_Iterator on second sequence (unused).

Parameters

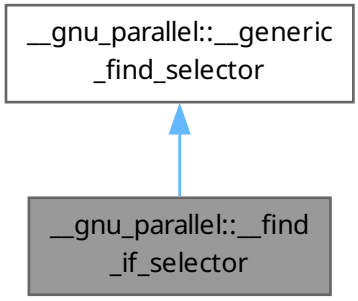
__pred	Find predicate.

The documentation for this struct was generated from the following file:

- find_selectors.h

5.17 __gnu_parallel::__find_if_selector Struct Reference

`#include <find_selectors.h>`
Inheritance diagram for __gnu_parallel::__find_if_selector:



Public Member Functions

- template<typename _RAIter1, typename _RAIter2, typename _Pred>
  std::pair< _RAIter1, _RAIter2 > _M_sequential_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)
- template<typename _RAIter1, typename _RAIter2, typename _Pred>
  bool operator() (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)

5.17.1 Detailed Description

Test predicate on a single element, used for std::find() and std::find_if ().

5.17.2 Member Function Documentation

_M_sequential_algorithm()

```
template<typename _RAIter1, typename _RAIter2, typename _Pred>
std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_if_selector::_M_sequential_algorithm (
            _RAIter1 __begin1,
            _RAIter1 __end1,
            _RAIter2 __begin2,
            _Pred __pred)  [inline]
```

Corresponding sequential algorithm on a sequence.

Parameters

__begin1	Begin iterator of first sequence.
__end1	End iterator of first sequence.
__begin2	Begin iterator of second sequence.
__pred	Find predicate.

References std::make_pair().

operator()()

```
template<typename _RAIter1, typename _RAIter2, typename _Pred>
bool __gnu_parallel::__find_if_selector::operator() (
            _RAIter1 __i1,
            _RAIter2 __i2,
            _Pred __pred)  [inline]
```

Test on one position.

Parameters

__i1	_Iterator on first sequence.
__i2	_Iterator on second sequence (unused).
__pred	Find predicate.

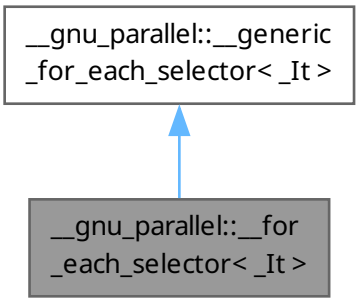
The documentation for this struct was generated from the following file:

- find_selectors.h

5.18 __gnu_parallel::__for_each_selector< _It > Struct Template Reference

```
#include <for_each_selectors.h>
```

Inheritance diagram for __gnu_parallel::__for_each_selector< _It >:



Public Member Functions

- template<typename _Op>
  bool operator() (_Op &__o, _It __i)

Public Attributes

- _It _M_finish_iterator

5.18.1 Detailed Description

template<typename _It>
struct __gnu_parallel::__for_each_selector< _It >

std::for_each() selector.

5.18.2 Member Function Documentation

operator()()

```
template<typename _It>
template<typename _Op>
bool __gnu_parallel::__for_each_selector< _It >::operator() (
            _Op & __o,
            _It __i)  [inline]
```

Functor execution.

Parameters

_↩ _o	Operator.
_↩ _i	iterator referencing object.

5.18.3 Member Data Documentation

_M_finish_iterator

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator  [inherited]
```

_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).
The documentation for this struct was generated from the following file:

- for_each_selectors.h

5.19 __cxxabiv1::__forced_unwind Class Reference

```
#include <cxxabi_forced.h>
```

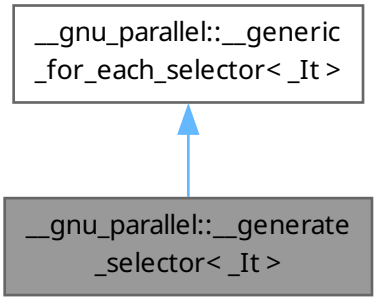
5.19.1 Detailed Description

Thrown as part of forced unwinding.
A magic placeholder class that can be caught by reference to recognize forced unwinding.
The documentation for this class was generated from the following file:

- cxxabi_forced.h

5.20 __gnu_parallel::__generate_selector< _It > Struct Template Reference

`#include <for_each_selectors.h>`
Inheritance diagram for __gnu_parallel::__generate_selector< _It >:



Public Member Functions

- template<typename _Op>
  bool operator() (_Op &__o, _It __i)

Public Attributes

- _It _M_finish_iterator

5.20.1 Detailed Description

template<typename _It>
struct __gnu_parallel::__generate_selector< _It >

std::generate() selector.

5.20.2 Member Function Documentation

operator()()

```
template<typename _It>
template<typename _Op>
bool __gnu_parallel::__generate_selector< _It >::operator() (
            _Op & __o,
            _It __i)  [inline]
```

Functor execution.

Parameters

_↩ _o	Operator.
_↩ _i	iterator referencing object.

5.20.3 Member Data Documentation

_M_finish_iterator

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator  [inherited]
```

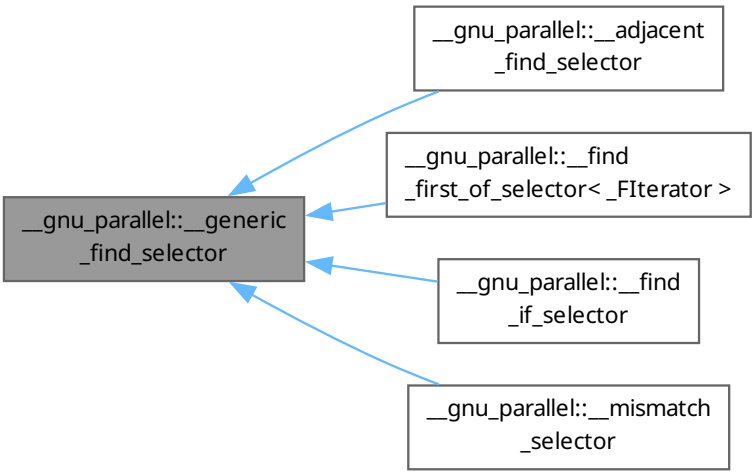
_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).
The documentation for this struct was generated from the following file:

- for_each_selectors.h

5.21 __gnu_parallel::__generic_find_selector Struct Reference

```
#include <find_selectors.h>
```

Inheritance diagram for __gnu_parallel::__generic_find_selector:



5.21.1 Detailed Description

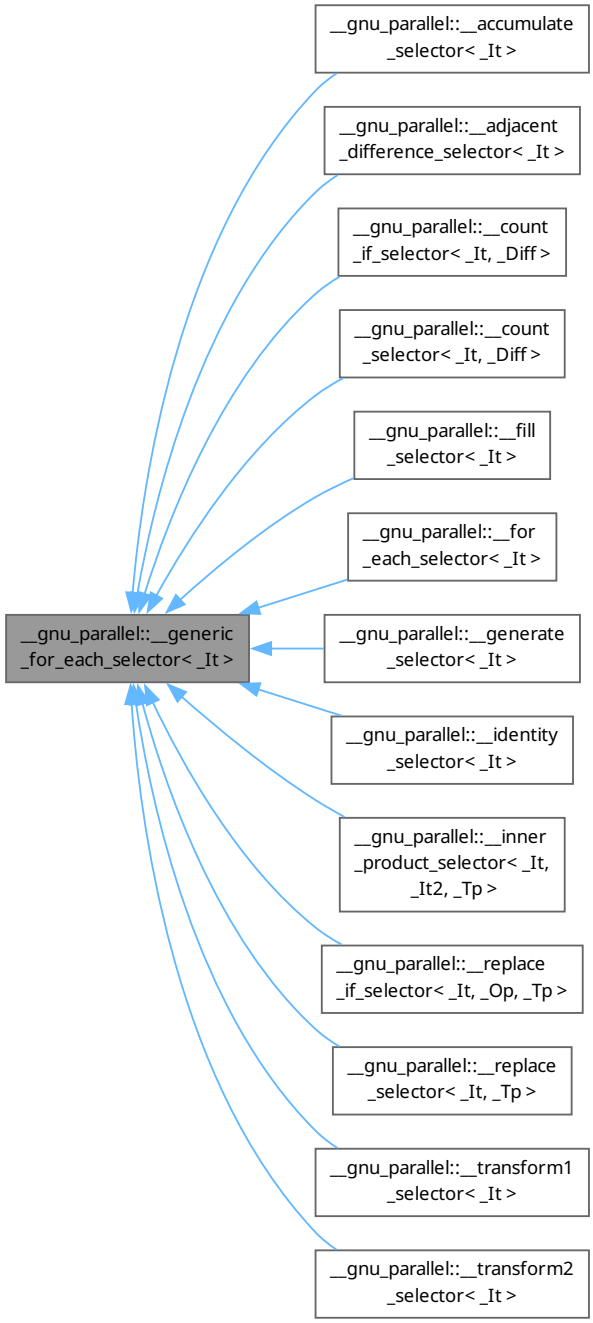
Base class of all __gnu_parallel::__find_template selectors.
The documentation for this struct was generated from the following file:

- find_selectors.h

5.22 __gnu_parallel::__generic_for_each_selector< _It > Struct Template Reference

```
#include <for_each_selectors.h>
```

Inheritance diagram for __gnu_parallel::__generic_for_each_selector< _It >:



Public Attributes

- _It _M_finish_iterator

5.22.1 Detailed Description

template<typename _It>
struct __gnu_parallel::__generic_for_each_selector< _It >

Generic __selector for embarrassingly parallel functions.

5.22.2 Member Data Documentation

_M_finish_iterator

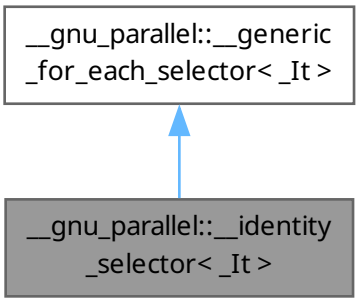
```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator
```

_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).
The documentation for this struct was generated from the following file:

- for_each_selectors.h

5.23 __gnu_parallel::__identity_selector< _It > Struct Template Reference

`#include <for_each_selectors.h>`
Inheritance diagram for __gnu_parallel::__identity_selector< _It >:



Public Member Functions

- template<typename _Op>
  _It operator() (_Op __o, _It __i)

Public Attributes

- _It _M_finish_iterator

5.23.1 Detailed Description

template<typename _It>
struct __gnu_parallel::__identity_selector< _It >

Selector that just returns the passed iterator.

5.23.2 Member Function Documentation

operator()()

```
template<typename _It>
template<typename _Op>
_It __gnu_parallel::__identity_selector< _It >::operator() (
            _Op __o,
            _It __i)  [inline]
```

Functor execution.

Parameters

_↩ _o	Operator (unused).
_↩ _i	iterator referencing object.

Returns

Passed iterator.

5.23.3 Member Data Documentation

_M_finish_iterator

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator  [inherited]
```

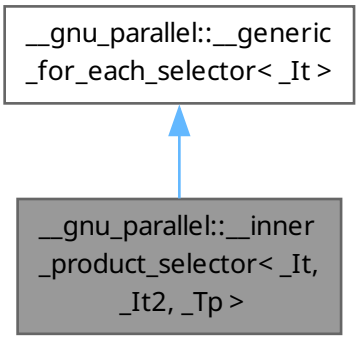
_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).
The documentation for this struct was generated from the following file:

- for_each_selectors.h

5.24 __gnu_parallel::__inner_product_selector< _It, _It2, _Tp > Struct Template Reference

```
#include <for_each_selectors.h>
```

Inheritance diagram for __gnu_parallel::__inner_product_selector< _It, _It2, _Tp >:



Public Member Functions

- __inner_product_selector (_It __b1, _It2 __b2)
- template<typename _Op>
  _Tp operator() (_Op __mult, _It __current)

Public Attributes

- _It __begin1_iterator
- _It2 __begin2_iterator
- _It _M_finish_iterator

5.24.1 Detailed Description

template<typename _It, typename _It2, typename _Tp>
struct __gnu_parallel::__inner_product_selector< _It, _It2, _Tp >

std::inner_product() selector.

5.24.2 Constructor & Destructor Documentation

__inner_product_selector()

```
template<typename _It, typename _It2, typename _Tp>
__gnu_parallel::__inner_product_selector< _It, _It2, _Tp >::__inner_product_selector (
            _It __b1,
            _It2 __b2)  [inline], [explicit]
```

Constructor.

Parameters

__b1	Begin iterator of first sequence.
__b2	Begin iterator of second sequence.

References __begin1_iterator, and __begin2_iterator.

5.24.3 Member Function Documentation

operator()()

```
template<typename _It, typename _It2, typename _Tp>
template<typename _Op>
_Tp __gnu_parallel::__inner_product_selector< _It, _It2, _Tp >::operator() (
            _Op __mult,
            _It __current)  [inline]
```

Functor execution.

Parameters

__mult	Multiplication functor.
__current	iterator referencing object.

Returns

Inner product elemental __result.

References __begin1_iterator, and __begin2_iterator.

5.24.4 Member Data Documentation

__begin1_iterator

```
template<typename _It, typename _It2, typename _Tp>
_It __gnu_parallel::__inner_product_selector< _It, _It2, _Tp >::__begin1_iterator
```

Begin iterator of first sequence.
Referenced by __inner_product_selector(), and operator()().

__begin2_iterator

```
template<typename _It, typename _It2, typename _Tp>
_It2 __gnu_parallel::__inner_product_selector< _It, _It2, _Tp >::__begin2_iterator
```

Begin iterator of second sequence.
Referenced by __inner_product_selector(), and operator()().

_M_finish_iterator

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator  [inherited]
```

_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).
The documentation for this struct was generated from the following file:

- for_each_selectors.h

5.25 std::__is_fast_hash< _Hash > Struct Template Reference

```
#include <functional_hash.h>
```

5.25.1 Detailed Description

template<typename _Hash>
struct std::__is_fast_hash< _Hash >

Hint about performance of hash functions.
If a given hash function object is not fast, the hash-based containers will cache the hash code. The default behavior is to consider that hashers are fast unless specified otherwise.
Users can specialize this for their own hash functions in order to force caching of hash codes in unordered containers. Specializing this trait affects the ABI of the unordered containers, so use it carefully.
The documentation for this struct was generated from the following file:

- functional_hash.h

5.26 std::__is_location_invariant< _Tp > Struct Template Reference

```
#include <std_function.h>
```

5.26.1 Detailed Description

template<typename _Tp>
struct std::__is_location_invariant< _Tp >

Trait identifying "location-invariant" types, meaning that the address of the object (or any of its members) will not escape. Trivially copyable types are location-invariant and users can specialize this trait for other types.
The documentation for this struct was generated from the following file:

- std_function.h

5.27 __gnu_parallel::__max_element_reduct< _Compare, _It > Struct Template Reference

`#include <for_each_selectors.h>`

Public Member Functions

- **__max_element_reduct** (_Compare &__c)
- _It **operator()** (_It __x, _It __y)

Public Attributes

- _Compare & **__comp**

5.27.1 Detailed Description

template<typename _Compare, typename _It>
struct __gnu_parallel::__max_element_reduct< _Compare, _It >

Reduction for finding the maximum element, using a comparator.
The documentation for this struct was generated from the following file:

- for_each_selectors.h

5.28 __gnu_parallel::__min_element_reduct< _Compare, _It > Struct Template Reference

`#include <for_each_selectors.h>`

Public Member Functions

- **__min_element_reduct** (_Compare &__c)
- _It **operator()** (_It __x, _It __y)

Public Attributes

- _Compare & **__comp**

5.28.1 Detailed Description

template<typename _Compare, typename _It>
struct __gnu_parallel::__min_element_reduct< _Compare, _It >

Reduction for finding the maximum element, using a comparator.
The documentation for this struct was generated from the following file:

- for_each_selectors.h

5.29 __gnu_cxx::__detail::__mini_vector< _Tp > Class Template Reference

`#include <bitmap_allocator.h>`

Public Types

- typedef const _Tp & **const_reference**
- typedef std::ptrdiff_t **difference_type**
- typedef pointer **iterator**
- typedef _Tp * **pointer**
- typedef _Tp & **reference**

- typedef std::size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- reference **back** () const throw ()
- iterator **begin** () const throw ()
- void **clear** () throw ()
- iterator **end** () const throw ()
- void **erase** (iterator __pos) throw ()
- void **insert** (iterator __pos, const_reference __x)
- reference **operator[ ]** (const size_type __pos) const throw ()
- void **pop_back** () throw ()
- void **push_back** (const_reference __x)
- size_type **size** () const throw ()

5.29.1 Detailed Description

template<**typename _Tp**>
class __gnu_cxx::__detail::__mini_vector< **_Tp** >

__mini_vector<> is a stripped down version of the full-fledged std::vector<>.
It is to be used only for built-in types or PODs. Notable differences are:

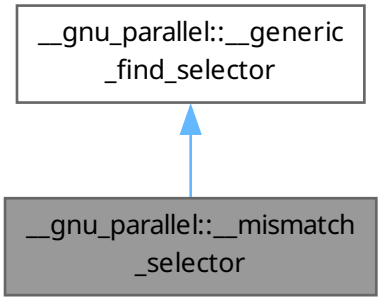
1. Not all accessor functions are present.
2. Used ONLY for PODs.
3. No Allocator template argument. Uses operator new() to get memory, and operator delete() to free it. Caveat: The dtor does NOT free the memory allocated, so this a memory-leaking vector!

The documentation for this class was generated from the following file:

- bitmap_allocator.h

5.30 __gnu_parallel::__mismatch_selector Struct Reference

`#include <find_selectors.h>`
Inheritance diagram for __gnu_parallel::__mismatch_selector:



Public Member Functions

- template<typename _RAIter1, typename _RAIter2, typename _Pred>
  std::pair< _RAIter1, _RAIter2 > _M_sequential_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)
- template<typename _RAIter1, typename _RAIter2, typename _Pred>
  bool operator() (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)

5.30.1 Detailed Description

Test inverted predicate on a single element.

5.30.2 Member Function Documentation

_M_sequential_algorithm()

```
template<typename _RAIter1, typename _RAIter2, typename _Pred>
std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__mismatch_selector::_M_sequential_algorithm (
            _RAIter1 __begin1,
            _RAIter1 __end1,
            _RAIter2 __begin2,
            _Pred __pred)  [inline]
```

Corresponding sequential algorithm on a sequence.

Parameters

__begin1	Begin iterator of first sequence.
__end1	End iterator of first sequence.
__begin2	Begin iterator of second sequence.
__pred	Find predicate.

operator()()

```
template<typename _RAIter1, typename _RAIter2, typename _Pred>
bool __gnu_parallel::__mismatch_selector::operator() (
            _RAIter1 __i1,
            _RAIter2 __i2,
            _Pred __pred)  [inline]
```

Test on one position.

Parameters

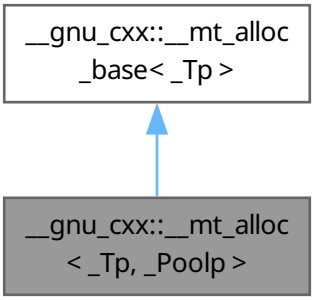
__i1	_Iterator on first sequence.
__i2	_Iterator on second sequence (unused).
__pred	Find predicate.

The documentation for this struct was generated from the following file:

- find_selectors.h

5.31 __gnu_cxx::__mt_alloc< _Tp, _Poolp > Class Template Reference

`#include <mt_allocator.h>`
Inheritance diagram for __gnu_cxx::__mt_alloc< _Tp, _Poolp >:



Public Types

- typedef _Poolp **__policy_type**
- typedef _Poolp::pool_type **__pool_type**
- typedef const _Tp * **const_pointer**
- typedef const _Tp & **const_reference**
- typedef std::ptrdiff_t **difference_type**
- typedef _Tp * **pointer**
- typedef std::true_type **propagate_on_container_move_assignment**
- typedef _Tp & **reference**
- typedef std::size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **__mt_alloc** (const __mt_alloc &) noexcept
- template<typename _Tp1, typename _Poolp1>
  __mt_alloc (const __mt_alloc< _Tp1, _Poolp1 > &) noexcept
- const __pool_base::_Tune **_M_get_options** ()
- void **_M_set_options** (__pool_base::_Tune __t)
- const_pointer **address** (const_reference __x) const noexcept
- pointer **address** (reference __x) const noexcept
- pointer **allocate** (size_type __n, const void *=0)
- template<typename _Up, typename... _Args>
  void **construct** (_Up *__p, _Args &&... __args)
- void **deallocate** (pointer __p, size_type __n)
- template<typename _Up>
  void **destroy** (_Up *__p)
- size_type **max_size** () const noexcept

5.31.1 Detailed Description

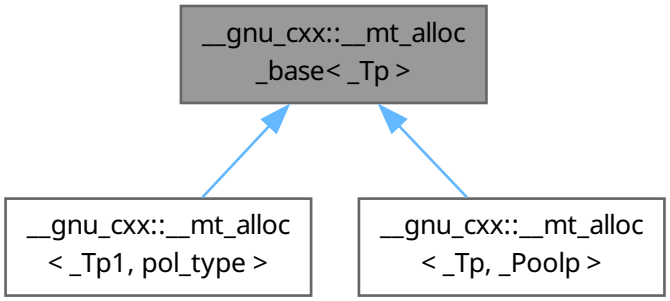
template<typename _Tp, typename _Poolp = __common_pool_policy<__pool, true >>
class __gnu_cxx::__mt_alloc< _Tp, _Poolp >

This is a fixed size (power of 2) allocator which - when compiled with thread support - will maintain one freelist per size per thread plus a *global* one. Steps are taken to limit the per thread freelist sizes (by returning excess back to the *global* list).
Further details: https://gcc.gnu.org/onlinedocs/libstdc++/manual/mt_allocator.html
The documentation for this class was generated from the following file:

- mt_allocator.h

5.32 __gnu_cxx::__mt_alloc_base< _Tp > Class Template Reference

`#include <mt_allocator.h>`
Inheritance diagram for __gnu_cxx::__mt_alloc_base< _Tp >:



Public Types

- typedef const _Tp * **const_pointer**
- typedef const _Tp & **const_reference**
- typedef std::ptrdiff_t **difference_type**
- typedef _Tp * **pointer**
- typedef std::true_type **propagate_on_container_move_assignment**
- typedef _Tp & **reference**
- typedef std::size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- const_pointer **address** (const_reference __x) const noexcept
- pointer **address** (reference __x) const noexcept
- template<typename _Up, typename... _Args>
  void **construct** (_Up *__p, _Args &&... __args)
- template<typename _Up>
  void **destroy** (_Up *__p)
- size_type **max_size** () const noexcept

5.32.1 Detailed Description

template<typename _Tp>
class __gnu_cxx::__mt_alloc_base< _Tp >

Base class for _Tp dependent member functions.
The documentation for this class was generated from the following file:

- mt_allocator.h

5.33 __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference

`#include <multiway_merge.h>`

Public Member Functions

- _RAIter3 **operator()** (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _↩ DifferenceTp __length, _Compare __comp)

5.33.1 Detailed Description

template<bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
struct __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >

Switch for 3-way merging with __sentinels turned off.
Note that 3-way merging is always stable!
The documentation for this struct was generated from the following file:

- multiway_merge.h

5.34 __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference

`#include <multiway_merge.h>`

Public Member Functions

- _RAIter3 **operator()** (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _↩ DifferenceTp __length, _Compare __comp)
- _RAIter3 **operator()** (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _↩ DifferenceTp __length, _Compare __comp)

5.34.1 Detailed Description

template<typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
struct __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _↩ DifferenceTp, _Compare >

Switch for 3-way merging with __sentinels turned on.
Note that 3-way merging is always stable!
The documentation for this struct was generated from the following file:

- multiway_merge.h

5.35 __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __sentinels, _RAlterIterator, _RAlter3, _DifferenceTp, _Compare > Struct Template Reference

`#include <multiway_merge.h>`

Public Member Functions

- _RAlter3 **operator()** (_RAlterIterator __seqs_begin, _RAlterIterator __seqs_end, _RAlter3 __target, _↩ DifferenceTp __length, _Compare __comp)

5.35.1 Detailed Description

template<bool __sentinels, typename _RAlterIterator, typename _RAlter3, typename _DifferenceTp, typename _Compare>
struct __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __sentinels, _RAlterIterator, _RAlter3, _DifferenceTp, _Compare >

Switch for 4-way merging with __sentinels turned off.
Note that 4-way merging is always stable!
The documentation for this struct was generated from the following file:

- multiway_merge.h

5.36 __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAlterIterator, _RAlter3, _DifferenceTp, _Compare > Struct Template Reference

`#include <multiway_merge.h>`

Public Member Functions

- _RAlter3 **operator()** (_RAlterIterator __seqs_begin, _RAlterIterator __seqs_end, _RAlter3 __target, _↩ DifferenceTp __length, _Compare __comp)
- _RAlter3 **operator()** (_RAlterIterator __seqs_begin, _RAlterIterator __seqs_end, _RAlter3 __target, _↩ DifferenceTp __length, _Compare __comp)

5.36.1 Detailed Description

template<typename _RAlterIterator, typename _RAlter3, typename _DifferenceTp, typename _Compare>
struct __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAlterIterator, _RAlter3, _↩ DifferenceTp, _Compare >

Switch for 4-way merging with __sentinels turned on.
Note that 4-way merging is always stable!
The documentation for this struct was generated from the following file:

- multiway_merge.h

5.37 __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable, _RAlterIterator, _RAlter3, _DifferenceTp, _Compare > Struct Template Reference

`#include <multiway_merge.h>`

Public Member Functions

- _RAIter3 **operator()** (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)

5.37.1 Detailed Description

template<bool __sentinels, bool __stable, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
struct __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >

Switch for k-way merging with __sentinels turned on.
The documentation for this struct was generated from the following file:

- multiway_merge.h

5.38 __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference

`#include <multiway_merge.h>`

Public Member Functions

- _RAIter3 **operator()** (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)
- _RAIter3 **operator()** (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)

5.38.1 Detailed Description

template<bool __stable, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
struct __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >

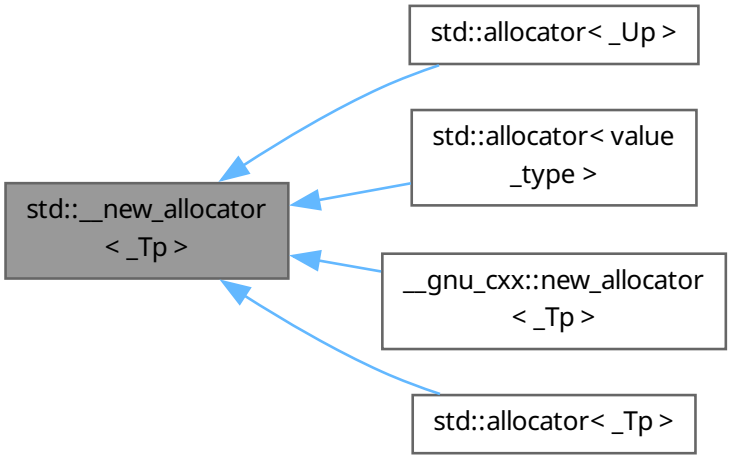
Switch for k-way merging with __sentinels turned off.
The documentation for this struct was generated from the following file:

- multiway_merge.h

5.39 std::__new_allocator< _Tp > Class Template Reference

`#include <memory>`

Inheritance diagram for std::__new_allocator< _Tp >:



Public Types

- typedef std::ptrdiff_t **difference_type**
- typedef std::true_type **propagate_on_container_move_assignment**
- typedef std::size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- constexpr **__new_allocator** (const __new_allocator &) noexcept
- template<typename _Tp1>
  constexpr **__new_allocator** (const __new_allocator< _Tp1 > &) noexcept
- _Tp * **allocate** (size_type __n, const void *=static_cast< const void * >(0))
- void **deallocate** (_Tp *__p, size_type __n)
- __new_allocator & **operator=** (const __new_allocator &)=default

Friends

- template<typename _Up>
  constexpr bool **operator==** (const __new_allocator &, const __new_allocator< _Up > &) noexcept

5.39.1 Detailed Description

template<typename _Tp>
class std::__new_allocator< _Tp >

An allocator that uses global `new`, as per C++03 [20.4.1].
This is precisely the allocator defined in the C++ Standard.

- all allocation calls `operator new`

- all deallocation calls `operator delete`

This is the default base-class implementation of `std::allocator`, and is also the base-class of the `__gnu_cxx::new_allocator` extension. You should use either `std::allocator` or `__gnu_cxx::new_allocator` instead of using this directly.

Template Parameters

_Tp	Type of allocated object.

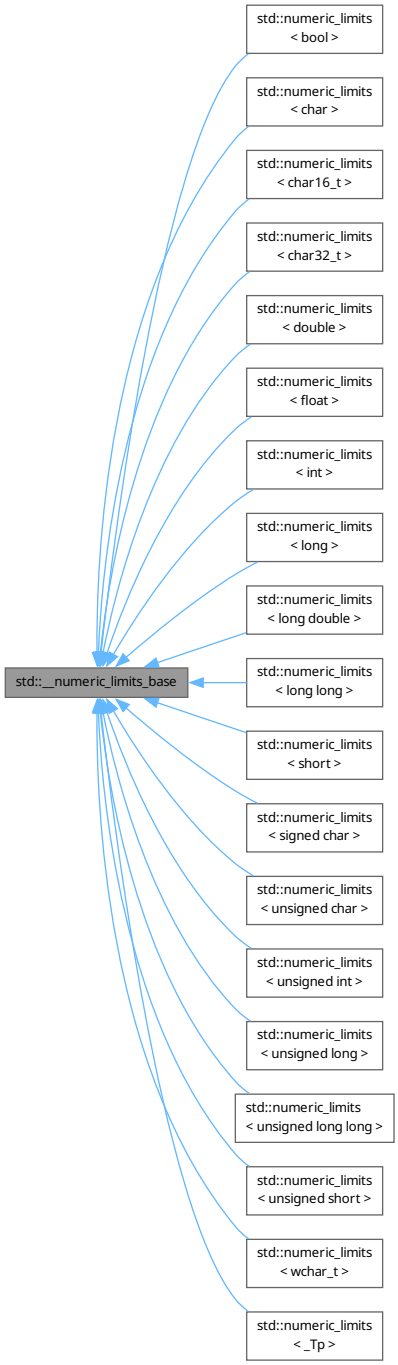
The documentation for this class was generated from the following file:

- bits/new_allocator.h

5.40 std::__numeric_limits_base Struct Reference

`#include <limits>`

Inheritance diagram for std::__numeric_limits_base:



Static Public Attributes

- static constexpr int digits
- static constexpr int digits10

- static constexpr float_denorm_style has_denorm
- static constexpr bool has_denorm_loss
- static constexpr bool has_infinity
- static constexpr bool has_quiet_NaN
- static constexpr bool has_signaling_NaN
- static constexpr bool is_bounded
- static constexpr bool is_exact
- static constexpr bool is_iec559
- static constexpr bool is_integer
- static constexpr bool is_modulo
- static constexpr bool is_signed
- static constexpr bool is_specialized
- static constexpr int max_digits10
- static constexpr int max_exponent
- static constexpr int max_exponent10
- static constexpr int min_exponent
- static constexpr int min_exponent10
- static constexpr int radix
- static constexpr float_round_style round_style
- static constexpr bool tinyness_before
- static constexpr bool traps

5.40.1 Detailed Description

Part of std::numeric_limits.
The `static const` members are usable as integral constant expressions.

Note

This is a separate class for purposes of efficiency; you should only access these members as part of an instantiation of the std::numeric_limits class.

5.40.2 Member Data Documentation

digits

`int std::__numeric_limits_base::digits [static], [constexpr]`
The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.
Referenced by std::generate_canonical(), std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::operator()(), std::operator<<(), and std::linear_congruential_engine< uint_fast32_t, 16807UL, 0UL, 2147483647UL >::seed().

digits10

`int std::__numeric_limits_base::digits10 [static], [constexpr]`
The number of base 10 digits that can be represented without change.

has_denorm

`float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]`
See std::float_denorm_style for more information.

has_denorm_loss

`bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]`
True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

has_infinity

`bool std::__numeric_limits_base::has_infinity [static], [constexpr]`
True if the type has a representation for positive infinity.

has_quiet_NaN

`bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]`
True if the type has a representation for a quiet (non-signaling) Not a Number.

has_signaling_NaN

`bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]`
True if the type has a representation for a signaling Not a Number.

is_bounded

`bool std::__numeric_limits_base::is_bounded [static], [constexpr]`
True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

is_exact

`bool std::__numeric_limits_base::is_exact [static], [constexpr]`
True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

is_iec559

`bool std::__numeric_limits_base::is_iec559 [static], [constexpr]`
True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

is_integer

`bool std::__numeric_limits_base::is_integer [static], [constexpr]`
True if the type is integer.

is_modulo

`bool std::__numeric_limits_base::is_modulo [static], [constexpr]`
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or $*$ on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

is_signed

`bool std::__numeric_limits_base::is_signed [static], [constexpr]`
True if the type is signed.

is_specialized

`bool std::__numeric_limits_base::is_specialized [static], [constexpr]`
This will be true for all fundamental types (which have specializations), and false for everything else.

max_digits10

`int std::__numeric_limits_base::max_digits10 [static], [constexpr]`
The number of base 10 digits required to ensure that values which differ are always differentiated.

max_exponent

`int std::__numeric_limits_base::max_exponent [static], [constexpr]`
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

max_exponent10

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr]`
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

min_exponent

`int std::__numeric_limits_base::min_exponent [static], [constexpr]`
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

min_exponent10

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr]`
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

radix

`int std::__numeric_limits_base::radix [static], [constexpr]`
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

round_style

`float_round_style std::__numeric_limits_base::round_style [static], [constexpr]`
See std::float_round_style for more information. This is only meaningful for floating types; integer types will all be round_toward_zero.

tinyness_before

`bool std::__numeric_limits_base::tinyness_before [static], [constexpr]`
True if tininess is detected before rounding. (see IEC 559)

traps

`bool std::__numeric_limits_base::traps [static], [constexpr]`
True if trapping is implemented for this type.
The documentation for this struct was generated from the following file:

- limits

5.41 __gnu_cxx::__per_type_pool_policy< _Tp, _PoolTp, _Thread > Struct Template Reference

`#include <mt_allocator.h>`

5.41.1 Detailed Description

template<typename _Tp, template< bool > class _PoolTp, bool _Thread>
struct __gnu_cxx::__per_type_pool_policy< _Tp, _PoolTp, _Thread >

Policy for individual __pool objects.
The documentation for this struct was generated from the following file:

- mt_allocator.h

5.42 __gnu_cxx::__pool< _Thread > Class Template Reference

5.42.1 Detailed Description

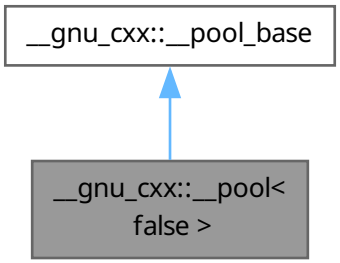
template<bool _Thread>
class __gnu_cxx::__pool< _Thread >

Data describing the underlying memory pool, parameterized on threading support.
The documentation for this class was generated from the following file:

- mt_allocator.h

5.43 __gnu_cxx::__pool< false > Class Reference

`#include <mt_allocator.h>`
Inheritance diagram for __gnu_cxx::__pool< false >:



Public Types

- typedef unsigned short int **_Binmap_type**
- typedef std::size_t **size_t**

Public Member Functions

- **__pool** (const __pool_base::_Tune &__tune)
- void **_M_adjust_freelist** (const _Bin_record &, _Block_record *, size_t)
- bool **_M_check_threshold** (size_t __bytes)
- void **_M_destroy** () throw ()
- size_t **_M_get_align** ()
- const _Bin_record & **_M_get_bin** (size_t __which)

- size_t **_M_get_binmap** (size_t __bytes)
- const _Tune & **_M_get_options** () const
- size_t **_M_get_thread_id** ()
- void **_M_initialize_once** ()
- void **_M_reclaim_block** (char *__p, size_t __bytes) throw ()
- char * **_M_reserve_block** (size_t __bytes, const size_t __thread_id)
- void **_M_set_options** (_Tune __t)

Protected Attributes

- _Binmap_type * **_M_binmap**
- bool **_M_init**
- _Tune **_M_options**

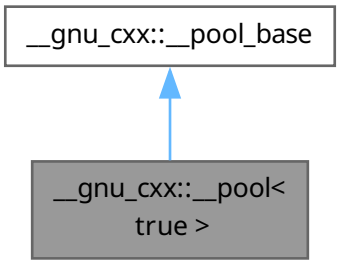
5.43.1 Detailed Description

Specialization for single thread.
The documentation for this class was generated from the following file:

- mt_allocator.h

5.44 __gnu_cxx::__pool< true > Class Reference

`#include <mt_allocator.h>`
Inheritance diagram for __gnu_cxx::__pool< true >:



Public Types

- typedef unsigned short int **_Binmap_type**
- typedef std::size_t **size_t**

Public Member Functions

- **__pool** (const __pool_base::_Tune &__tune)
- void **_M_adjust_freelist** (const _Bin_record &__bin, _Block_record *__block, size_t __thread_id)
- bool **_M_check_threshold** (size_t __bytes)
- void **_M_destroy** () throw ()

- void **_M_destroy_thread_key** (void *) throw ()
- size_t **_M_get_align** ()
- const _Bin_record & **_M_get_bin** (size_t __which)
- size_t **_M_get_binmap** (size_t __bytes)
- const _Tune & **_M_get_options** () const
- size_t **_M_get_thread_id** ()
- void **_M_initialize** (__destroy_handler)
- void **_M_initialize_once** ()
- void **_M_reclaim_block** (char *__p, size_t __bytes) throw ()
- char * **_M_reserve_block** (size_t __bytes, const size_t __thread_id)
- void **_M_set_options** (_Tune __t)

Protected Attributes

- _Binmap_type * **_M_binmap**
- bool **_M_init**
- _Tune **_M_options**

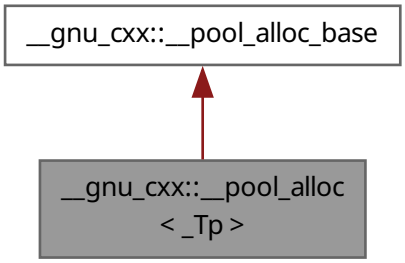
5.44.1 Detailed Description

Specialization for thread enabled, via gthreads.h.
The documentation for this class was generated from the following file:

- mt_allocator.h

5.45 __gnu_cxx::__pool_alloc< _Tp > Class Template Reference

`#include <pool_allocator.h>`
Inheritance diagram for __gnu_cxx::__pool_alloc< _Tp >:



Public Types

- typedef const _Tp * **const_pointer**
- typedef const _Tp & **const_reference**
- typedef std::ptrdiff_t **difference_type**
- typedef _Tp * **pointer**
- typedef std::true_type **propagate_on_container_move_assignment**

- typedef _Tp & **reference**
- typedef std::size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **__pool_alloc** (const __pool_alloc &) noexcept
- template<typename _Tp1>
  __pool_alloc (const __pool_alloc< _Tp1 > &) noexcept
- const_pointer **address** (const_reference __x) const noexcept
- pointer **address** (reference __x) const noexcept
- pointer **allocate** (size_type __n, const void *=0)
- template<typename _Up, typename... _Args>
  void **construct** (_Up *__p, _Args &&... __args)
- void **deallocate** (pointer __p, size_type __n)
- template<typename _Up>
  void **destroy** (_Up *__p)
- size_type **max_size** () const noexcept

5.45.1 Detailed Description

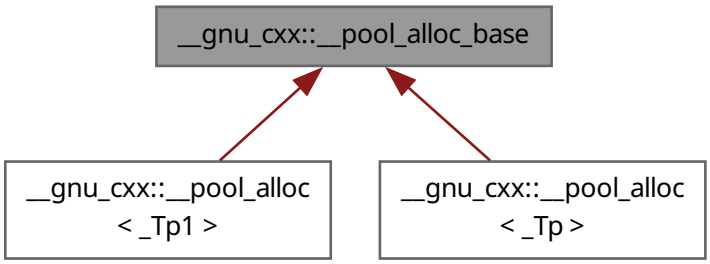
template<typename _Tp>
class __gnu_cxx::__pool_alloc< _Tp >

Allocator using a memory pool with a single lock.
The documentation for this class was generated from the following file:

- pool_allocator.h

5.46 __gnu_cxx::__pool_alloc_base Class Reference

`#include <pool_allocator.h>`
Inheritance diagram for __gnu_cxx::__pool_alloc_base:



Protected Types

- enum { **_S_free_list_size** }
- enum { **_S_max_bytes** }
- enum { **_S_align** }

Protected Member Functions

- char * **_M_allocate_chunk** (size_t __n, int &__nobjs)
- _Obj *volatile * **_M_get_free_list** (size_t __bytes) throw ()
- __mutex & **_M_get_mutex** () throw ()
- void * **_M_refill** (size_t __n)
- size_t **_M_round_up** (size_t __bytes)

Static Protected Attributes

- static char * **_S_end_free**
- static _Obj *volatile **_S_free_list** [_S_free_list_size]
- static size_t **_S_heap_size**
- static char * **_S_start_free**

5.46.1 Detailed Description

Base class for __pool_alloc.
Uses various allocators to fulfill underlying requests (and makes as few requests as possible when in default high-speed pool mode).
Important implementation properties:

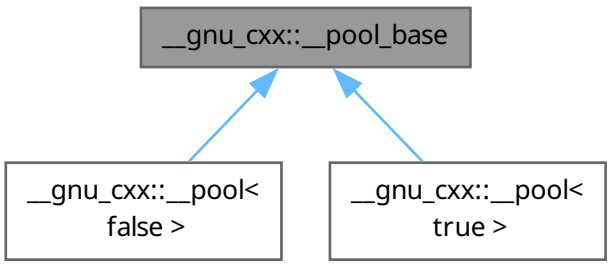
1. If globally mandated, then allocate objects from new
2. If the clients request an object of size > _S_max_bytes, the resulting object will be obtained directly from new
3. In all other cases, we allocate an object of size exactly _S_round_up(requested_size). Thus the client has enough size information that we can return the object to the proper free list without permanently losing part of the object.

The documentation for this class was generated from the following file:

- pool_allocator.h

5.47 __gnu_cxx::__pool_base Struct Reference

`#include <mt_allocator.h>`
Inheritance diagram for __gnu_cxx::__pool_base:



Public Types

- typedef unsigned short int **_Binmap_type**
- typedef std::size_t **size_t**

Public Member Functions

- **__pool_base** (const _Tune &__options)
- bool **_M_check_threshold** (size_t __bytes)
- size_t **_M_get_align** ()
- size_t **_M_get_binmap** (size_t __bytes)
- const _Tune & **_M_get_options** () const
- void **_M_set_options** (_Tune __t)

Protected Attributes

- _Binmap_type * **_M_binmap**
- bool **_M_init**
- _Tune **_M_options**

5.47.1 Detailed Description

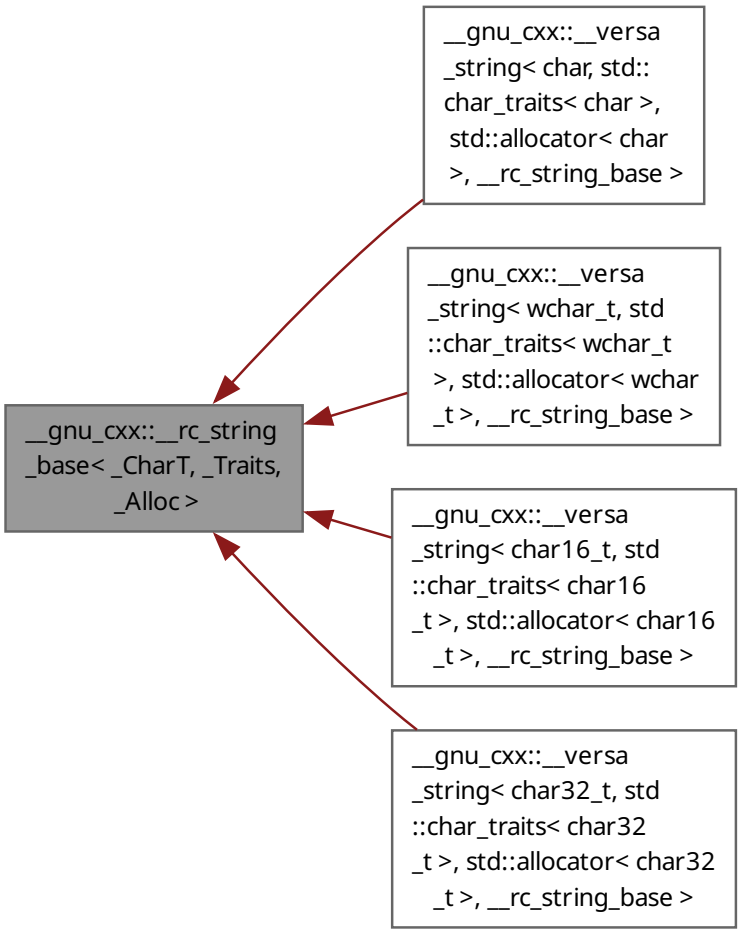
Base class for pool object.
The documentation for this struct was generated from the following file:

- mt_allocator.h

5.48 __gnu_cxx::__rc_string_base< _CharT, _Traits, _Alloc > Class Template Reference

`#include <rc_string_base.h>`

Inheritance diagram for __gnu_cxx::__rc_string_base< _CharT, _Traits, _Alloc >:



Public Types

- typedef _Util_Base::_CharT_alloc_type **_CharT_alloc_type**
- typedef __vstring_utility< _CharT, _Traits, _Alloc > **_Util_Base**
- typedef _Alloc **allocator_type**
- typedef _CharT_alloc_type::size_type **size_type**
- typedef _Traits **traits_type**
- typedef _Traits::char_type **value_type**

Public Member Functions

- **__rc_string_base** (__rc_string_base &&__rcs)
- template<typename _InputIterator>
  __rc_string_base (_InputIterator __beg, _InputIterator __end, const _Alloc &__a)
- **__rc_string_base** (const __rc_string_base &__rcs)

- **__rc_string_base** (const _Alloc &__a)
- **__rc_string_base** (size_type __n, _CharT __c, const _Alloc &__a)
- void **_M_assign** (const __rc_string_base &__rcs)
- size_type **_M_capacity** () const
- void **_M_clear** ()
- bool **_M_compare** (const __rc_string_base &) const
- bool **_M_compare** (const __rc_string_base &__rcs) const
- bool **_M_compare** (const __rc_string_base &__rcs) const
- _CharT * **_M_data** () const
- void **_M_erase** (size_type __pos, size_type __n)
- allocator_type & **_M_get_allocator** ()
- const allocator_type & **_M_get_allocator** () const
- bool **_M_is_shared** () const
- void **_M_leak** ()
- size_type **_M_length** () const
- size_type **_M_max_size** () const
- void **_M_mutate** (size_type __pos, size_type __len1, const _CharT *__s, size_type __len2)
- void **_M_reserve** (size_type __res)
- void **_M_set_leaked** ()
- void **_M_set_length** (size_type __n)
- void **_M_swap** (__rc_string_base &__rcs)
- template<typename _InIterator>
  _CharT * **_S_construct** (_InIterator __beg, _InIterator __end, const _Alloc &__a, std::forward_iterator_tag)

Protected Types

- typedef __gnu_cxx::__normal_iterator< const_pointer, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, __rc_string_base > > **__const_rc_iterator**
- typedef __gnu_cxx::__normal_iterator< const_pointer, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _↩
  _sso_string_base > > **__const_sso_iterator**
- typedef __gnu_cxx::__normal_iterator< pointer, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, __rc_string_base > > **__rc_iterator**
- typedef __gnu_cxx::__normal_iterator< pointer, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, __sso_↩
  string_base > > **__sso_iterator**
- typedef __alloc_traits< _CharT_alloc_type > **_CharT_alloc_traits**
- typedef _CharT_alloc_traits::const_pointer **const_pointer**
- typedef _CharT_alloc_type::difference_type **difference_type**
- typedef _CharT_alloc_traits::pointer **pointer**

Static Protected Member Functions

- static void **_S_assign** (_CharT *__d, size_type __n, _CharT __c)
- static int **_S_compare** (size_type __n1, size_type __n2)
- static void **_S_copy** (_CharT *__d, const _CharT *__s, size_type __n)
- static void **_S_copy_chars** (_CharT *__p, __const_rc_iterator __k1, __const_rc_iterator __k2)
- static void **_S_copy_chars** (_CharT *__p, __const_sso_iterator __k1, __const_sso_iterator __k2)
- static void **_S_copy_chars** (_CharT *__p, __rc_iterator __k1, __rc_iterator __k2)
- static void **_S_copy_chars** (_CharT *__p, __sso_iterator __k1, __sso_iterator __k2)
- static void **_S_copy_chars** (_CharT *__p, _CharT *__k1, _CharT *__k2)
- template<typename _Iterator>
  static void **_S_copy_chars** (_CharT *__p, _Iterator __k1, _Iterator __k2)
- static void **_S_copy_chars** (_CharT *__p, const _CharT *__k1, const _CharT *__k2)
- static void **_S_move** (_CharT *__d, const _CharT *__s, size_type __n)

5.48.1 Detailed Description

template<typename _CharT, typename _Traits, typename _Alloc>
class __gnu_cxx::__rc_string_base< _CharT, _Traits, _Alloc >

Documentation? What's that? Nathan Myers ncm@cantrip.org.
A string looks like this:

```
                                    [_Rep]
                                    _M_length
[__rc_string_base<char_type>]       _M_capacity
_M_dataplus                         _M_refcount
_M_p ---------------->              unnamed array of char_type
```

Where the _M_p points to the first character in the string, and you cast it to a pointer-to-_Rep and subtract 1 to get a pointer to the header.
This approach has the enormous advantage that a string object requires only one allocation. All the ugliness is confined within a single pair of inline functions, which each compile to a single *add* instruction: _Rep::_M_refdata(), and __rc_↩ string_base::_M_rep(); and the allocation function which gets a block of raw bytes and with room enough and constructs a _Rep object at the front.
The reason you want _M_data pointing to the character array and not the _Rep is so that the debugger can see the string contents. (Probably we should add a non-inline member to get the _Rep for the debugger to use, so users can check the actual string length.)
Note that the _Rep object is a POD so that you can have a static *empty string* _Rep object already *constructed* before static constructors have run. The reference-count encoding is chosen so that a 0 indicates one reference, so you never try to destroy the empty-string _Rep object.
All but the last paragraph is considered pretty conventional for a C++ string implementation.
The documentation for this class was generated from the following file:

- rc_string_base.h

5.49 std::tr2::__reflection_typelist< _Elements > Struct Template Reference

5.49.1 Detailed Description

template<typename... _Elements>
struct std::tr2::__reflection_typelist< _Elements >

See N2965: Type traits and base classes by Michael Spertus Simple typelist. Compile-time list of types.
The documentation for this struct was generated from the following file:

- tr2/type_traits

5.50 std::tr2::__reflection_typelist< _First, _Rest... > Struct Template Reference

`#include <type_traits>`

Public Types

- typedef std::false_type **empty**

5.50.1 Detailed Description

template<typename _First, typename... _Rest>
struct std::tr2::__reflection_typelist< _First, _Rest... >

Partial specialization.
The documentation for this struct was generated from the following file:

- tr2/type_traits

5.51 std::tr2::__reflection_typelist<> Struct Reference

`#include <type_traits>`

Public Types

- typedef std::true_type **empty**

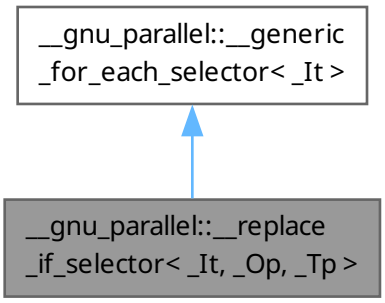
5.51.1 Detailed Description

Specialization for an empty typelist.
The documentation for this struct was generated from the following file:

- tr2/type_traits

5.52 __gnu_parallel::__replace_if_selector< _It, _Op, _Tp > Struct Template Reference

`#include <for_each_selectors.h>`
Inheritance diagram for __gnu_parallel::__replace_if_selector< _It, _Op, _Tp >:



Public Member Functions

- __replace_if_selector (const _Tp &__new_val)
- bool operator() (_Op &__o, _It __i)

Public Attributes

- const _Tp & __new_val
- _It _M_finish_iterator

5.52.1 Detailed Description

template<typename _It, typename _Op, typename _Tp>
struct __gnu_parallel::__replace_if_selector< _It, _Op, _Tp >

std::replace() selector.

5.52.2 Constructor & Destructor Documentation

__replace_if_selector()

```
template<typename _It, typename _Op, typename _Tp>
__gnu_parallel::__replace_if_selector< _It, _Op, _Tp >::__replace_if_selector (
            const _Tp & __new_val)  [inline], [explicit]
```

Constructor.

Parameters

__new_val	Value to replace with.

References __new_val.

5.52.3 Member Function Documentation

operator()()

```
template<typename _It, typename _Op, typename _Tp>
bool __gnu_parallel::__replace_if_selector< _It, _Op, _Tp >::operator() (
            _Op & __o,
            _It __i)  [inline]
```

Functor execution.

Parameters

_↩ _o	Operator.
_↩ _i	iterator referencing object.

References __new_val.

5.52.4 Member Data Documentation

__new_val

```
template<typename _It, typename _Op, typename _Tp>
const _Tp& __gnu_parallel::__replace_if_selector< _It, _Op, _Tp >::__new_val
```

Value to replace with.
Referenced by __replace_if_selector(), and operator()().

_M_finish_iterator

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator  [inherited]
```

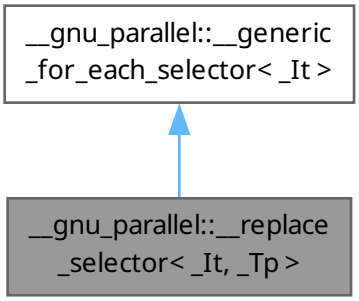
_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).
The documentation for this struct was generated from the following file:

- for_each_selectors.h

5.53 __gnu_parallel::__replace_selector< _It, _Tp > Struct Template Reference

```
#include <for_each_selectors.h>
```

Inheritance diagram for __gnu_parallel::__replace_selector< _It, _Tp >:



Public Member Functions

- __replace_selector (const _Tp &__new_val)
- bool operator() (_Tp &__v, _It __i)

Public Attributes

- const _Tp & __new_val
- _It _M_finish_iterator

5.53.1 Detailed Description

template<typename _It, typename _Tp>
struct __gnu_parallel::__replace_selector< _It, _Tp >

std::replace() selector.

5.53.2 Constructor & Destructor Documentation

__replace_selector()

```
template<typename _It, typename _Tp>
__gnu_parallel::__replace_selector< _It, _Tp >::__replace_selector (
            const _Tp & __new_val)  [inline], [explicit]
```

Constructor.

Parameters

__new_val	Value to replace with.

References __new_val.

5.53.3 Member Function Documentation

operator()()

```
template<typename _It, typename _Tp>
bool __gnu_parallel::__replace_selector< _It, _Tp >::operator() (
            _Tp & __v,
            _It __i)  [inline]
```

Functor execution.

Parameters

_↩ _v	Current value.
_↩ _i	iterator referencing object.

References __new_val.

5.53.4 Member Data Documentation

__new_val

```
template<typename _It, typename _Tp>
const _Tp& __gnu_parallel::__replace_selector< _It, _Tp >::__new_val
```

Value to replace with.
Referenced by __replace_selector(), and operator()().

_M_finish_iterator

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator  [inherited]
```

_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).
The documentation for this struct was generated from the following file:

- for_each_selectors.h

5.54 __gnu_cxx::__scoped_lock Class Reference

```
#include <concurrence.h>
```

Public Types

- typedef __mutex **__mutex_type**

Public Member Functions

- **__scoped_lock** (__mutex_type &__name)

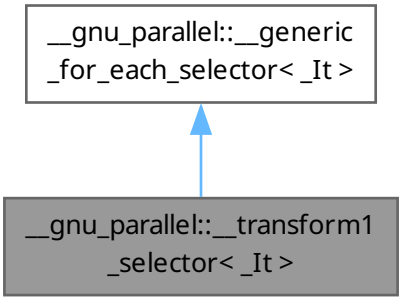
5.54.1 Detailed Description

Scoped lock idiom.
The documentation for this class was generated from the following file:

- concurrence.h

5.55 __gnu_parallel::__transform1_selector< _It > Struct Template Reference

`#include <for_each_selectors.h>`
Inheritance diagram for __gnu_parallel::__transform1_selector< _It >:



Public Member Functions

- template<typename _Op>
  bool operator() (_Op &__o, _It __i)

Public Attributes

- _It _M_finish_iterator

5.55.1 Detailed Description

template<typename _It>
struct __gnu_parallel::__transform1_selector< _It >

std::transform() __selector, one input sequence variant.

5.55.2 Member Function Documentation

operator()()

```
template<typename _It>
template<typename _Op>
bool __gnu_parallel::__transform1_selector< _It >::operator() (
            _Op & __o,
            _It __i)  [inline]
```

Functor execution.

Parameters

_↩ _o	Operator.
_↩ _i	iterator referencing object.

5.55.3 Member Data Documentation

_M_finish_iterator

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator  [inherited]
```

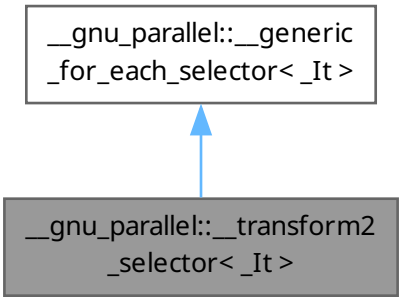
_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).
The documentation for this struct was generated from the following file:

- for_each_selectors.h

5.56 __gnu_parallel::__transform2_selector< _It > Struct Template Reference

```
#include <for_each_selectors.h>
```

Inheritance diagram for __gnu_parallel::__transform2_selector< _It >:



Public Member Functions

- template<typename _Op>
  bool operator() (_Op &__o, _It __i)

Public Attributes

- _It _M_finish_iterator

5.56.1 Detailed Description

template<typename _It>
struct __gnu_parallel::__transform2_selector< _It >

std::transform() __selector, two input sequences variant.

5.56.2 Member Function Documentation

operator()()

```
template<typename _It>
template<typename _Op>
bool __gnu_parallel::__transform2_selector< _It >::operator() (
```

```
        _Op & __o,
        _It __i)  [inline]
```

Functor execution.

Parameters

_↩ _o	Operator.
_↩ _i	iterator referencing object.

5.56.3 Member Data Documentation

_M_finish_iterator

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator  [inherited]
```

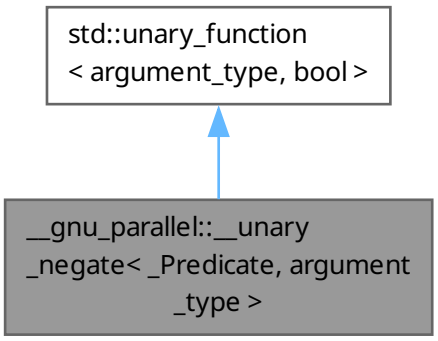
_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).
The documentation for this struct was generated from the following file:

- for_each_selectors.h

5.57 __gnu_parallel::__unary_negate< _Predicate, argument_type > Class Template Reference

```
#include <base.h>
```

Inheritance diagram for __gnu_parallel::__unary_negate< _Predicate, argument_type >:



Public Types

- typedef argument_type argument_type
- typedef bool result_type

Public Member Functions

- **__unary_negate** (const _Predicate &__x)
- bool **operator()** (const argument_type &__x)

Protected Attributes

- _Predicate **_M_pred**

5.57.1 Detailed Description

template<typename _Predicate, typename argument_type>
class __gnu_parallel::__unary_negate< _Predicate, argument_type >

Similar to std::unary_negate, but giving the argument types explicitly.

5.57.2 Member Typedef Documentation

argument_type

```
typedef argument_type std::unary_function< argument_type, bool >::argument_type  [inherited]
```

`argument_type` is the type of the argument

result_type

```
typedef bool std::unary_function< argument_type, bool >::result_type  [inherited]
```

`result_type` is the return type
The documentation for this class was generated from the following file:

- base.h

5.58 __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > Class Template Reference

```
#include <vstring.h>
```

Public Types

- typedef _Alloc **allocator_type**
- typedef __gnu_cxx::__normal_iterator< const_pointer, __versa_string > **const_iterator**
- typedef _CharT_alloc_traits::const_pointer **const_pointer**
- typedef const value_type & **const_reference**
- typedef std::reverse_iterator< const_iterator > **const_reverse_iterator**
- typedef _CharT_alloc_type::difference_type **difference_type**
- typedef __gnu_cxx::__normal_iterator< pointer, __versa_string > **iterator**
- typedef _CharT_alloc_traits::pointer **pointer**
- typedef value_type & **reference**
- typedef std::reverse_iterator< iterator > **reverse_iterator**
- typedef _CharT_alloc_type::size_type **size_type**
- typedef _Traits **traits_type**
- typedef _Traits::char_type **value_type**

Public Member Functions

- __versa_string (__versa_string &&__str) noexcept
- template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  __versa_string (_InputIterator __beg, _InputIterator __end, const _Alloc &__a=_Alloc())
- __versa_string (const __versa_string &__str)
- __versa_string (const __versa_string &__str, size_type __pos, size_type __n, const _Alloc &__a)
- __versa_string (const __versa_string &__str, size_type __pos, size_type __n=npos)
- __versa_string (const _Alloc &__a=_Alloc()) noexcept
- __versa_string (const _CharT *__s, const _Alloc &__a=_Alloc())
- __versa_string (const _CharT *__s, size_type __n, const _Alloc &__a=_Alloc())
- __versa_string (size_type __n, _CharT __c, const _Alloc &__a=_Alloc())
- __versa_string (std::initializer_list< _CharT > __l, const _Alloc &__a=_Alloc())
- ~__versa_string () noexcept
- template<typename _InputIterator>
  __versa_string< _CharT, _Traits, _Alloc, _Base > & **_M_replace_dispatch** (const_iterator __i1, const_iterator __i2, _InputIterator __k1, _InputIterator __k2, std::__false_type)
- template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  __versa_string & append (_InputIterator __first, _InputIterator __last)
- __versa_string & append (const __versa_string &__str)
- __versa_string & append (const __versa_string &__str, size_type __pos, size_type __n)
- __versa_string & append (const _CharT *__s)
- __versa_string & append (const _CharT *__s, size_type __n)
- __versa_string & append (size_type __n, _CharT __c)
- __versa_string & append (std::initializer_list< _CharT > __l)
- __versa_string & assign (__versa_string &&__str) noexcept
- template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  __versa_string & assign (_InputIterator __first, _InputIterator __last)
- __versa_string & assign (const __versa_string &__str)
- __versa_string & assign (const __versa_string &__str, size_type __pos, size_type __n)
- __versa_string & assign (const _CharT *__s)
- __versa_string & assign (const _CharT *__s, size_type __n)
- __versa_string & assign (size_type __n, _CharT __c)
- __versa_string & assign (std::initializer_list< _CharT > __l)
- reference at (size_type __n)
- const_reference at (size_type __n) const
- const_reference back () const noexcept
- reference back () noexcept
- const_iterator begin () const noexcept
- iterator begin () noexcept
- const _CharT * c_str () const noexcept
- size_type capacity () const noexcept
- const_iterator cbegin () const noexcept
- const_iterator cend () const noexcept
- void clear () noexcept
- int compare (const __versa_string &__str) const
- int compare (const _CharT *__s) const
- int compare (size_type __pos, size_type __n, const __versa_string &__str) const
- int compare (size_type __pos, size_type __n1, const _CharT *__s) const
- int compare (size_type __pos, size_type __n1, const _CharT *__s, size_type __n2) const
- int compare (size_type __pos1, size_type __n1, const __versa_string &__str, size_type __pos2, size_type __n2) const

- size_type copy (_CharT *__s, size_type __n, size_type __pos=0) const
- const_reverse_iterator crbegin () const noexcept
- const_reverse_iterator crend () const noexcept
- const _CharT * data () const noexcept
- bool empty () const noexcept
- const_iterator end () const noexcept
- iterator end () noexcept
- iterator erase (const_iterator __first, const_iterator __last)
- iterator erase (const_iterator __position)
- __versa_string & erase (size_type __pos=0, size_type __n=npos)
- size_type find (_CharT __c, size_type __pos=0) const noexcept
- size_type find (const __versa_string &__str, size_type __pos=0) const noexcept
- size_type find (const _CharT *__s, size_type __pos, size_type __n) const
- size_type find (const _CharT *__s, size_type __pos=0) const
- size_type find_first_not_of (_CharT __c, size_type __pos=0) const noexcept
- size_type find_first_not_of (const __versa_string &__str, size_type __pos=0) const noexcept
- size_type find_first_not_of (const _CharT *__s, size_type __pos, size_type __n) const
- size_type find_first_not_of (const _CharT *__s, size_type __pos=0) const
- size_type find_first_of (_CharT __c, size_type __pos=0) const noexcept
- size_type find_first_of (const __versa_string &__str, size_type __pos=0) const noexcept
- size_type find_first_of (const _CharT *__s, size_type __pos, size_type __n) const
- size_type find_first_of (const _CharT *__s, size_type __pos=0) const
- size_type find_last_not_of (_CharT __c, size_type __pos=npos) const noexcept
- size_type find_last_not_of (const __versa_string &__str, size_type __pos=npos) const noexcept
- size_type find_last_not_of (const _CharT *__s, size_type __pos, size_type __n) const
- size_type find_last_not_of (const _CharT *__s, size_type __pos=npos) const
- size_type find_last_of (_CharT __c, size_type __pos=npos) const noexcept
- size_type find_last_of (const __versa_string &__str, size_type __pos=npos) const noexcept
- size_type find_last_of (const _CharT *__s, size_type __pos, size_type __n) const
- size_type find_last_of (const _CharT *__s, size_type __pos=npos) const
- const_reference front () const noexcept
- reference front () noexcept
- allocator_type get_allocator () const noexcept
- iterator insert (const_iterator __p, _CharT __c)
- template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  iterator insert (const_iterator __p, _InputIterator __beg, _InputIterator __end)
- iterator insert (const_iterator __p, size_type __n, _CharT __c)
- iterator insert (const_iterator __p, std::initializer_list< _CharT > __l)
- __versa_string & insert (size_type __pos, const _CharT *__s)
- __versa_string & insert (size_type __pos, const _CharT *__s, size_type __n)
- __versa_string & insert (size_type __pos, size_type __n, _CharT __c)
- __versa_string & insert (size_type __pos1, const __versa_string &__str)
- __versa_string & insert (size_type __pos1, const __versa_string &__str, size_type __pos2, size_type __n)
- size_type length () const noexcept
- size_type max_size () const noexcept
- __versa_string & operator+= (_CharT __c)
- __versa_string & operator+= (const __versa_string &__str)
- __versa_string & operator+= (const _CharT *__s)
- __versa_string & operator+= (std::initializer_list< _CharT > __l)
- __versa_string & operator= (__versa_string &&__str) noexcept
- __versa_string & operator= (_CharT __c)

- __versa_string & operator= (const __versa_string &__str)
- __versa_string & operator= (const _CharT *__s)
- __versa_string & operator= (std::initializer_list< _CharT > __l)
- const_reference operator[ ] (size_type __pos) const noexcept
- reference operator[ ] (size_type __pos) noexcept
- void pop_back ()
- void push_back (_CharT __c)
- const_reverse_iterator rbegin () const noexcept
- reverse_iterator rbegin () noexcept
- const_reverse_iterator rend () const noexcept
- reverse_iterator rend () noexcept
- __versa_string & **replace** (const_iterator __i1, const_iterator __i2, _CharT *__k1, _CharT *__k2)
- template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  __versa_string & replace (const_iterator __i1, const_iterator __i2, _InputIterator __k1, _InputIterator __k2)
- __versa_string & replace (const_iterator __i1, const_iterator __i2, const __versa_string &__str)
- __versa_string & **replace** (const_iterator __i1, const_iterator __i2, const _CharT *__k1, const _CharT *__k2)
- __versa_string & replace (const_iterator __i1, const_iterator __i2, const _CharT *__s)
- __versa_string & replace (const_iterator __i1, const_iterator __i2, const _CharT *__s, size_type __n)
- __versa_string & **replace** (const_iterator __i1, const_iterator __i2, const_iterator __k1, const_iterator __k2)
- __versa_string & **replace** (const_iterator __i1, const_iterator __i2, iterator __k1, iterator __k2)
- __versa_string & replace (const_iterator __i1, const_iterator __i2, size_type __n, _CharT __c)
- __versa_string & replace (const_iterator __i1, const_iterator __i2, std::initializer_list< _CharT > __l)
- __versa_string & replace (size_type __pos, size_type __n, const __versa_string &__str)
- __versa_string & replace (size_type __pos, size_type __n1, const _CharT *__s)
- __versa_string & replace (size_type __pos, size_type __n1, const _CharT *__s, size_type __n2)
- __versa_string & replace (size_type __pos, size_type __n1, size_type __n2, _CharT __c)
- __versa_string & replace (size_type __pos1, size_type __n1, const __versa_string &__str, size_type __pos2, size_type __n2)
- void reserve (size_type __res_arg=0)
- void resize (size_type __n)
- void resize (size_type __n, _CharT __c)
- size_type rfind (_CharT __c, size_type __pos=npos) const noexcept
- size_type rfind (const __versa_string &__str, size_type __pos=npos) const noexcept
- size_type rfind (const _CharT *__s, size_type __pos, size_type __n) const
- size_type rfind (const _CharT *__s, size_type __pos=npos) const
- void shrink_to_fit () noexcept
- size_type size () const noexcept
- __versa_string substr (size_type __pos=0, size_type __n=npos) const
- void swap (__versa_string &__s) noexcept

Static Public Attributes

- static const size_type npos

5.58.1 Detailed Description

template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
class __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >

Template class __versa_string.
Data structure managing sequences of characters and character-like objects.

5.58.2 Constructor & Destructor Documentation

__versa_string() [1/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
            const _Alloc & __a = _Alloc())  [inline], [explicit], [noexcept]
```

Construct an empty string using allocator *a*.

Referenced by compare(), compare(), and __gnu_cxx::__versa_string< char >::substr().

__versa_string() [2/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str)  [inline]
```

Construct string with copy of value of *__str*.

Parameters

__str	Source string.

__versa_string() [3/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
            __versa_string< _CharT, _Traits, _Alloc, _Base > && __str)  [inline], [noexcept]
```

String move constructor.

Parameters

__str	Source string.

The newly-constructed string contains the exact contents of *__str*. The contents of *__str* are a valid, but unspecified string.

__versa_string() [4/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
            std::initializer_list< _CharT > __l,
            const _Alloc & __a = _Alloc())  [inline]
```

Construct string from an initializer list.

Parameters

_↩ _l	std::initializer_list of characters.
_↩ _a	Allocator to use (default is default allocator).

__versa_string() [5/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
            size_type __pos,
            size_type __n = npos)  [inline]
```

Construct string as copy of a substring.

Parameters

__str	Source string.
__pos	Index of first character to copy from.
__n	Number of characters to copy (default remainder).

__versa_string() [6/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
            size_type __pos,
            size_type __n,
            const _Alloc & __a)  [inline]
```

Construct string as copy of a substring.

Parameters

__str	Source string.
__pos	Index of first character to copy from.
__n	Number of characters to copy.
__a	Allocator to use.

__versa_string() [7/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
            const _CharT * __s,
            size_type __n,
            const _Alloc & __a = _Alloc())  [inline]
```

Construct string initialized by a character array.

Parameters

_↩ _s	Source character array.
_↩ _n	Number of characters to copy.
_↩ _a	Allocator to use (default is default allocator).

NB: __*s* must have at least __*n* characters, '\0' has no special meaning.

__versa_string() [8/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
            const _CharT * __s,
            const _Alloc & __a = _Alloc())  [inline]
```

Construct string as copy of a C string.

Parameters

_↩ _*s*	Source C string.
_↩ _*a*	Allocator to use (default is default allocator).

__versa_string() [9/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
            size_type __n,
            _CharT __c,
            const _Alloc & __a = _Alloc())  [inline]
```

Construct string as multiple characters.

Parameters

_↩ _*n*	Number of characters.
_↩ _*c*	Character to use.
_↩ _*a*	Allocator to use (default is default allocator).

__versa_string() [10/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
            _InputIterator __beg,
            _InputIterator __end,
            const _Alloc & __a = _Alloc())  [inline]
```

Construct string as copy of a range.

Parameters

__beg	Start of range.
__end	End of range.
__a	Allocator to use (default is default allocator).

~__versa_string()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::~__versa_string ()  [inline], [noexcept]
```

Destroy the string instance.

5.58.3 Member Function Documentation

append() [1/7]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (
            _InputIterator __first,
            _InputIterator __last)  [inline]
```

Append a range of characters.

Parameters

__first	Iterator referencing the first character to append.
__last	Iterator marking the end of the range.

Returns

Reference to this string.

Appends characters in the range [first,last) to this string.

append() [2/7]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str)  [inline]
```

Append a string to this string.

Parameters

__str	The string to append.

Returns

Reference to this string.

Referenced by __gnu_cxx::__versa_string< char >::append(), std::getline(), __gnu_cxx::operator+(), __gnu_cxx::operator+(), __gnu_cxx::operator+(), __gnu_cxx::__versa_string< char >::operator+=(), __gnu_cxx::__versa_string< char >::operator+=(), __gnu_cxx::__versa_string< char >::operator+=(), and resize().

append() [3/7]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
            size_type __pos,
            size_type __n)  [inline]
```

Append a substring.

Parameters

__str	The string to append.
__pos	Index of the first character of str to append.
__n	The number of characters to append.

Returns

Reference to this string.

Exceptions

std::out_of_range	if *pos* is not a valid index.

This function appends *__n* characters from *__str* starting at *__pos* to this string. If *__n* is is larger than the number of available characters in *__str*, the remainder of *__str* is appended.

append() [4/7]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (
            const _CharT * __s)  [inline]
```

Append a C string.

Parameters

__s	The C string to append.

Returns

Reference to this string.

append() [5/7]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (
            const _CharT * __s,
            size_type __n)  [inline]
```

Append a C substring.

Parameters

_↩ _s	The C string to append.
_↩ _n	The number of characters to append.

Returns

Reference to this string.

append() [6/7]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (
            size_type __n,
            _CharT __c)  [inline]
```

Append multiple characters.

Parameters

_↩ _n	The number of characters to append.
_↩ _c	The character to use.

Returns

Reference to this string.

Appends n copies of c to this string.

append() [7/7]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (
            std::initializer_list< _CharT > __l)  [inline]
```

Append an initializer_list of characters.

Parameters

↩ _↩ ↩ _↩ l	The initializer_list of characters to append.

Returns

Reference to this string.

assign() [1/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (
            __versa_string< _CharT, _Traits, _Alloc, _Base > && __str)  [inline], [noexcept]
```

Set value to contents of another string.

Parameters

__str	Source string to use.

Returns

Reference to this string.

This function sets this string to the exact contents of *__str*. *__str* is a valid, but unspecified string.

assign() [2/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (
            _InputIterator __first,
            _InputIterator __last)  [inline]
```

Set value to a range of characters.

Parameters

__first	Iterator referencing the first character to append.
__last	Iterator marking the end of the range.

Returns

Reference to this string.

Sets value of string to characters in the range [first,last).

assign() [3/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str)  [inline]
```

Set value to contents of another string.

Parameters

__str	Source string to use.

Returns

Reference to this string.

Referenced by __gnu_cxx::__versa_string< char >::assign(), __gnu_cxx::__versa_string< char >::operator=(), __gnu_cxx::__versa_string< char >::operator=(), __gnu_cxx::__versa_string< char >::operator=(), and __gnu_cxx::__versa_string< ch

assign() [4/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
            size_type __pos,
            size_type __n)  [inline]
```

Set value to a substring of a string.

Parameters

__str	The string to use.
__pos	Index of the first character of str.
__n	Number of characters to use.

Returns

Reference to this string.

Exceptions

std::out_of_range	if *__pos* is not a valid index.

This function sets this string to the substring of *__str* consisting of *__n* characters at *__pos*. If *__n* is is larger than the number of available characters in *__str*, the remainder of *__str* is used.

assign() [5/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (
            const _CharT * __s)  [inline]
```

Set value to contents of a C string.

Parameters

__s	The C string to use.

Returns

Reference to this string.

This function sets the value of this string to the value of *__s*. The data is copied, so there is no dependence on *__s* once the function returns.

assign() [6/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (
            const _CharT * __s,
            size_type __n)  [inline]
```

Set value to a C substring.

Parameters

_↩ _s	The C string to use.
_↩ _n	Number of characters to use.

Returns

Reference to this string.

This function sets the value of this string to the first __n characters of __s. If __n is is larger than the number of available characters in __s, the remainder of __s is used.

assign() [7/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (
            size_type __n,
            _CharT __c)  [inline]
```

Set value to multiple characters.

Parameters

_↩ _n	Length of the resulting string.
_↩ _c	The character to use.

Returns

Reference to this string.

This function sets the value of this string to __n copies of character __c.

assign() [8/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (
            std::initializer_list< _CharT > __l)  [inline]
```

Set value to an initializer_list of characters.

Parameters

↩ _↩ ↩ _↩ l	The initializer_list of characters to assign.

Returns

Reference to this string.

at() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::at (
            size_type __n)  [inline]
```

Provides access to the data contained in the string.

Parameters

_↩ _n	The index of the character to access.

Returns

Read/write reference to the character.

Exceptions

std::out_of_range	If *__n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws out_of_range if the check fails. Success results in unsharing the string.

at() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::at (
            size_type __n) const  [inline]
```

Provides access to the data contained in the string.

Parameters

_↩ _n	The index of the character to access.

Returns

Read-only (const) reference to the character.

Exceptions

std::out_of_range	If *__n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws out_of_range if the check fails.

back() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::back () const  [inline],
[noexcept]
```

Returns a read-only (constant) reference to the data at the last element of the string.

back() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::back () [inline], [noexcept]
```

Returns a read/write reference to the data at the last element of the string.

begin() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::begin () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first character in the string.

begin() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::begin () [inline], [noexcept]
```

Returns a read/write iterator that points to the first character in the string. Unshares the string.

Referenced by __gnu_cxx::__versa_string< char >::crend(), __gnu_cxx::__versa_string< char >::rend(), and __gnu_cxx::__versa_string< char >::rend().

c_str()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const _CharT * __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::c_str () const [inline],
[noexcept]
```

Return const pointer to null-terminated contents.

This is a handle to internal data. Do not modify or dire things may happen.

capacity()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::capacity () const [inline],
[noexcept]
```

Returns the total number of characters that the string can hold before needing to allocate more memory.

Referenced by __gnu_cxx::__versa_string< char >::push_back(), and __gnu_cxx::__versa_string< char >::shrink_to_fit().

cbegin()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::cbegin () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first character in the string.

cend()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::cend () const [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last character in the string.

clear()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::clear ()  [inline], [noexcept]
```

Erases the string, making it empty.

compare() [1/6]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str) const  [inline]
```

Compare to a string.

Parameters

__str	String to compare against.

Returns

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before *__str*, 0 if their values are equivalent, or > 0 if this string is ordered after *__str*. Determines the effective length rlen of the strings to compare as the smallest of size() and str.size(). The function then compares the two strings by calling traits::compare(data(), str.data(),rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.
Referenced by __gnu_cxx::operator<(), __gnu_cxx::operator<(), __gnu_cxx::operator<(), __gnu_cxx::operator<=(), __gnu_cxx::operator<=(), __gnu_cxx::operator<=(), __gnu_cxx::operator>(), __gnu_cxx::operator>(), __gnu_cxx::operator>(), __gnu_cxx::operator>=(), __gnu_cxx::operator>=(), and __gnu_cxx::operator>=().

compare() [2/6]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare (
            const _CharT * __s) const
```

Compare to a C string.

Parameters

__s	C string to compare against.

Returns

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before *__s*, 0 if their values are equivalent, or > 0 if this string is ordered after *__s*. Determines the effective length rlen of the strings to compare as the smallest of size() and the length of a string constructed from *__s*. The function then compares the two strings by calling traits::compare(data(),s,rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.
References std::min(), and size().

compare() [3/6]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare (
            size_type __pos,
            size_type __n,
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str) const
```

Compare substring to a string.

Parameters

__pos	Index of first character of substring.
__n	Number of characters in substring.
__str	String to compare against.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n* characters starting at *__pos*. Returns an integer < 0 if the substring is ordered before *__str*, 0 if their values are equivalent, or > 0 if the substring is ordered after *__str*. Determines the effective length rlen of the strings to compare as the smallest of the length of the substring and *__str.size()*. The function then compares the two strings by calling traits::compare(substring.data(),str.data(),rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.
References __versa_string(), data(), std::min(), and size().

compare() [4/6]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare (
            size_type __pos,
            size_type __n1,
            const _CharT * __s) const
```

Compare substring to a C string.

Parameters

__pos	Index of first character of substring.
__n1	Number of characters in substring.
__s	C string to compare against.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n1* characters starting at *__pos*. Returns an integer < 0 if the substring is ordered before *__s*, 0 if their values are equivalent, or > 0 if the substring is ordered after *__s*. Determines the effective length rlen of the strings to compare as the smallest of the length of the substring and the length of a string constructed from *__s*. The function then compares the two string by calling traits::compare(substring.data(),s,rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.
References std::min().

compare() [5/6]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare (
             size_type __pos,
             size_type __n1,
             const _CharT * __s,
             size_type __n2) const
```

Compare substring against a character array.

Parameters

__pos	Index of first character of substring.
__n1	Number of characters in substring.
__s	character array to compare against.
__n2	Number of characters of s.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n1* characters starting at *__pos*. Form a string from the first *__n2* characters of *__s*. Returns an integer < 0 if this substring is ordered before the string from *__s*, 0 if their values are equivalent, or > 0 if this substring is ordered after the string from *__s*. Determines the effective length rlen of the strings to compare as the smallest of the length of the substring and *__n2*. The function then compares the two strings by calling traits↩ ::compare(substring.data(),__s,rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.
NB: __s must have at least n2 characters, *\0* has no special meaning.
References std::min().

compare() [6/6]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare (
             size_type __pos1,
             size_type __n1,
             const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
             size_type __pos2,
             size_type __n2) const
```

Compare substring to a substring.

Parameters

__pos1	Index of first character of substring.
__n1	Number of characters in substring.
__str	String to compare against.
__pos2	Index of first character of substring of str.
__n2	Number of characters in substring of str.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n1* characters starting at *__pos1*. Form the substring of *__str* from the *__n2* characters starting at *__pos2*. Returns an integer < 0 if this substring is ordered before the substring of *__str*, 0 if their values are equivalent, or > 0 if this substring is ordered after the substring of *__str*. Determines the effective length rlen of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.
References __versa_string(), data(), and std::min().

copy()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↩
_Traits, _Alloc, _Base >::copy (
            _CharT * __s,
            size_type __n,
            size_type __pos = 0) const
```

Copy substring into C string.

Parameters

__s	C string to copy value into.
__n	Number of characters to copy.
__pos	Index of first character to copy.

Returns

Number of characters actually copied

Exceptions

std::out_of_range	If pos > size().

Copies up to *__n* characters starting at *__pos* into the C string *s*. If *__pos* is greater than size(), out_of_range is thrown.

crbegin()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::crbegin ()
const  [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

crend()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
const_reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::crend ()
const  [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

data()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const _CharT * __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::data () const  [inline],
[noexcept]
```

Return const pointer to contents.

This is a handle to internal data. Do not modify or dire things may happen.

Referenced by compare(), compare(), __gnu_cxx::__versa_string< char >::compare(), __gnu_cxx::__versa_string< char >::find(), __gnu_cxx::__versa_string< char >::find_first_not_of(), __gnu_cxx::__versa_string< char >::find_first_of(), __gnu_cxx::__versa_string< __gnu_cxx::operator==(), and __gnu_cxx::operator==().

empty()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::empty () const  [inline], [nodiscard],
[noexcept]
```

Returns true if the string is empty. Equivalent to `*this == ""`.

end() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::end () const  [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last character in the string.

end() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::end ()  [inline], [noexcept]
```

Returns a read/write iterator that points one past the last character in the string. Unshares the string.

Referenced by __gnu_cxx::__versa_string< char >::crbegin(), __gnu_cxx::__versa_string< char >::rbegin(), and __gnu_cxx::__versa_string< char >::rbegin().

erase() [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::erase (
            const_iterator __first,
            const_iterator __last)  [inline]
```

Remove a range of characters.

Parameters

__first	Iterator referencing the first character to remove.
__last	Iterator referencing the end of the range.

Returns

Iterator referencing location of first after removal.

Removes the characters in the range [first,last) from this string. The value of the string doesn't change if an error is thrown.

erase() [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::erase (
            const_iterator __position)  [inline]
```

Remove one character.

Parameters

__position	Iterator referencing the character to remove.

Returns

iterator referencing same location after removal.

Removes the character at *__position* from this string. The value of the string doesn't change if an error is thrown.

erase() [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::erase (
            size_type __pos = 0,
            size_type __n = npos)  [inline]
```

Remove characters.

Parameters

__pos	Index of first character to remove (default 0).
__n	Number of characters to remove (default remainder).

Returns

Reference to this string.

Exceptions

std::out_of_range	If *__pos* is beyond the end of this string.

Removes *__n* characters from this string starting at *__pos*. The length of the string is reduced by *__n*. If there are < *__n* characters to remove, the remainder of the string is truncated. If *__p* is beyond end of string, out_of_range is thrown. The value of the string doesn't change if an error is thrown.
Referenced by std::getline().

find() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↩
_Traits, _Alloc, _Base >::find (
            _CharT __c,
            size_type __pos = 0) const  [noexcept]
```

Find position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for *__c* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and size().

find() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
            size_type __pos = 0) const  [inline], [noexcept]
```

Find position of a string.

Parameters

__str	String to locate.
__pos	Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from *__pos*, searches forward for value of *__str* within this string. If found, returns the index where it begins. If not found, returns npos.

find() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↩
_Traits, _Alloc, _Base >::find (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const
```

Find position of a C substring.

Parameters

__s	C string to locate.
__pos	Index of character to search from.
__n	Number of characters from *__s* to search for.

Returns

Index of start of first occurrence.

Starting from *__pos*, searches forward for the first *__n* characters in *__s* within this string. If found, returns the index where it begins. If not found, returns npos.
References npos, and size().
Referenced by __gnu_cxx::__versa_string< char >::find(), __gnu_cxx::__versa_string< char >::find(), and __gnu_cxx::__versa_string<

find() [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find (
            const _CharT * __s,
            size_type __pos = 0) const  [inline]
```

Find position of a C string.

Parameters

__s	C string to locate.
__pos	Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from *__pos*, searches forward for the value of *__s* within this string. If found, returns the index where it begins. If not found, returns npos.

find_first_not_of() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↩
_Traits, _Alloc, _Base >::find_first_not_of (
            _CharT __c,
            size_type __pos = 0) const  [noexcept]
```

Find position of a different character.

Parameters

__c	Character to avoid.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character other than *__c* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and size().

find_first_not_of() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_not_of (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
            size_type __pos = 0) const  [inline], [noexcept]
```

Find position of a character not in string.

Parameters

__str	String containing characters to avoid.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character not contained in *__str* within this string. If found, returns the index where it was found. If not found, returns npos.
Referenced by __gnu_cxx::__versa_string< char >::find_first_not_of(), and __gnu_cxx::__versa_string< char >::find_first_not_of().

find_first_not_of() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↩
_Traits, _Alloc, _Base >::find_first_not_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const
```

Find position of a character not in C substring.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search from.
__n	Number of characters from s to consider.

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character not contained in the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and size().

find_first_not_of() [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_not_of (
            const _CharT * __s,
            size_type __pos = 0) const  [inline]
```

Find position of a character not in C string.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character not contained in *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_of() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of (
            _CharT __c,
            size_type __pos = 0) const  [inline], [noexcept]
```

Find position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for the character *__c* within this string. If found, returns the index where it was found. If not found, returns npos.
Note: equivalent to find(c, pos).

find_first_of() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
            size_type __pos = 0) const  [inline], [noexcept]
```

Find position of a character of string.

Parameters

__str	String containing characters to locate.

__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for one of the characters of *__str* within this string. If found, returns the index where it was found. If not found, returns npos.
Referenced by __gnu_cxx::__versa_string< char >::find_first_of(), and __gnu_cxx::__versa_string< char >::find_first_of().

find_first_of() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↩
_Traits, _Alloc, _Base >::find_first_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const
```

Find position of a character of C substring.

Parameters

__s	String containing characters to locate.
__pos	Index of character to search from.
__n	Number of characters from s to search for.

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for one of the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and size().

find_first_of() [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of (
            const _CharT * __s,
            size_type __pos = 0) const  [inline]
```

Find position of a character of C string.

Parameters

__s	String containing characters to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for one of the characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_not_of() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↩
_Traits, _Alloc, _Base >::find_last_not_of (
            _CharT __c,
            size_type __pos = npos) const  [noexcept]
```

Find last position of a different character.

Parameters

__c	Character to avoid.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character other than *__c* within this string. If found, returns the index where it was found. If not found, returns npos.
References size().

find_last_not_of() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_not_of (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
            size_type __pos = npos) const  [inline], [noexcept]
```

Find last position of a character not in string.

Parameters

__str	String containing characters to avoid.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character not contained in *__str* within this string. If found, returns the index where it was found. If not found, returns npos.
Referenced by __gnu_cxx::__versa_string< char >::find_last_not_of(), and __gnu_cxx::__versa_string< char >::find_last_not_of().

find_last_not_of() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↩
_Traits, _Alloc, _Base >::find_last_not_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const
```

Find last position of a character not in C substring.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search back from.
__n	Number of characters from s to consider.

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character not contained in the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and size().

find_last_not_of() [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_not_of (
            const _CharT * __s,
            size_type __pos = npos) const  [inline]
```

Find last position of a character not in C string.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character not contained in *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_of() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of (
            _CharT __c,
            size_type __pos = npos) const  [inline], [noexcept]
```

Find last position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for *__c* within this string. If found, returns the index where it was found. If not found, returns npos.
Note: equivalent to rfind(c, pos).

find_last_of() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
            size_type __pos = npos) const  [inline], [noexcept]
```

Find last position of a character of string.

Parameters

__str	String containing characters to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for one of the characters of *__str* within this string. If found, returns the index where it was found. If not found, returns npos.
Referenced by __gnu_cxx::__versa_string< char >::find_last_of(), and __gnu_cxx::__versa_string< char >::find_last_of().

find_last_of() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↩
_Traits, _Alloc, _Base >::find_last_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const
```

Find last position of a character of C substring.

Parameters

__s	C string containing characters to locate.
__pos	Index of character to search back from.
__n	Number of characters from s to search for.

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for one of the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and size().

find_last_of() [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of (
            const _CharT * __s,
            size_type __pos = npos) const  [inline]
```

Find last position of a character of C string.

Parameters

__s	C string containing characters to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for one of the characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

front() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::front () const  [inline],
[noexcept]
```

Returns a read-only (constant) reference to the data at the first element of the string.

front() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::front ()  [inline], [noexcept]
```

Returns a read/write reference to the data at the first element of the string.

get_allocator()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
allocator_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::get_allocator ()
const  [inline], [noexcept]
```

Return copy of allocator used to construct this string.

insert() [1/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
            const_iterator __p,
            _CharT __c)  [inline]
```

Insert one character.

Parameters

_↩ _p	Iterator referencing position in string to insert at.
_↩ _c	The character to insert.

Returns

Iterator referencing newly inserted char.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts character *__c* at position referenced by *__p*. If adding character causes the length to exceed max_size(), length_error is thrown. If *__p* is beyond end of string, out_of_range is thrown. The value of the string doesn't change if an error is thrown.

insert() [2/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
            const_iterator __p,
            _InputIterator __beg,
            _InputIterator __end)  [inline]
```

Insert a range of characters.

Parameters

__p	Const_iterator referencing location in string to insert at.
__beg	Start of range.
__end	End of range.

Returns

Iterator referencing the first inserted char.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts characters in range [beg,end). If adding characters causes the length to exceed max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

insert() [3/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
            const_iterator __p,
            size_type __n,
            _CharT __c)  [inline]
```

Insert multiple characters.

Parameters

__p	Const_iterator referencing location in string to insert at.
__n	Number of characters to insert

_↩ _c	The character to insert.

Returns

Iterator referencing the first inserted char.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts __n copies of character __c starting at the position referenced by iterator __p. If adding characters causes the length to exceed max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown. Referenced by __gnu_cxx::__versa_string< char >::insert().

insert() [4/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
            const_iterator __p,
            std::initializer_list< _CharT > __l)  [inline]
```

Insert an initializer_list of characters.

Parameters

_↩ _p	Const_iterator referencing location in string to insert at.
_↩ _l	The initializer_list of characters to insert.

Returns

Iterator referencing the first inserted char.

Exceptions

std::length_error	If new length exceeds `max_size()`.

insert() [5/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
            size_type __pos,
            const _CharT * __s)  [inline]
```

Insert a C string.

Parameters

__pos	Iterator referencing location in string to insert at.
__s	The C string to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.
std::out_of_range	If *__pos* is beyond the end of this string.

Inserts the first *__n* characters of *__s* starting at *__pos*. If adding characters causes the length to exceed max_size(), length_error is thrown. If *__pos* is beyond end(), out_of_range is thrown. The value of the string doesn't change if an error is thrown.

insert() [6/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
            size_type __pos,
            const _CharT * __s,
            size_type __n)  [inline]
```

Insert a C substring.

Parameters

__pos	Iterator referencing location in string to insert at.
__s	The C string to insert.
__n	The number of characters to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.
std::out_of_range	If *__pos* is beyond the end of this string.

Inserts the first *__n* characters of *__s* starting at *__pos*. If adding characters causes the length to exceed max_size(), length_error is thrown. If *__pos* is beyond end(), out_of_range is thrown. The value of the string doesn't change if an error is thrown.

insert() [7/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
            size_type __pos,
            size_type __n,
            _CharT __c)  [inline]
```

Insert multiple characters.

Parameters

__pos	Index in string to insert at.
__n	Number of characters to insert
__c	The character to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.
std::out_of_range	If *__pos* is beyond the end of this string.

Inserts *__n* copies of character *__c* starting at index *__pos*. If adding characters causes the length to exceed max_↩ size(), length_error is thrown. If *__pos* > length(), out_of_range is thrown. The value of the string doesn't change if an error is thrown.

insert() [8/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
            size_type __pos1,
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str)  [inline]
```

Insert value of a string.

Parameters

__pos1	Iterator referencing location in string to insert at.
__str	The string to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts value of *__str* starting at *__pos1*. If adding characters causes the length to exceed max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

insert() [9/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
            size_type __pos1,
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
            size_type __pos2,
            size_type __n)  [inline]
```

Insert a substring.

Parameters

__pos1	Iterator referencing location in string to insert at.
__str	The string to insert.
__pos2	Start of characters in str to insert.
__n	Number of characters to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.
std::out_of_range	If *__pos1* > size() or *__pos2* > *__str.size()*.

Starting at *__pos1*, insert *__n* character of *__str* beginning with *__pos2*. If adding characters causes the length to exceed max_size(), length_error is thrown. If *__pos1* is beyond the end of this string or *__pos2* is beyond the end of *__str*, out_of_range is thrown. The value of the string doesn't change if an error is thrown.

length()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::length () const  [inline],
[noexcept]
```

Returns the number of characters in the string, not including any null-termination.

max_size()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::max_size () const  [inline],
[noexcept]
```

Returns the size() of the largest possible string.
Referenced by std::getline().

operator+=() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator+= (
_CharT __c) [inline]

Append a character.

Parameters

_↩ _c	The character to append.

Returns

Reference to this string.

operator+=() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator+= (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str)  [inline]
```

Append a string to this string.

Parameters

__str	The string to append.

Returns

Reference to this string.

operator+=() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator+= (
            const _CharT * __s)  [inline]
```

Append a C string.

Parameters

_↩ _s	The C string to append.

Returns

Reference to this string.

operator+=() [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator+= (
            std::initializer_list< _CharT > __l)  [inline]
```

Append an initializer_list of characters.

Parameters

↩ _↩ ↩ _↩ *l*	The initializer_list of characters to be appended.

Returns

Reference to this string.

operator=() [1/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator= (
            __versa_string< _CharT, _Traits, _Alloc, _Base > && __str)  [inline], [noexcept]
```

String move assignment operator.

Parameters

__str	Source string.

The contents of *__str* are moved into this string (without copying). *__str* is a valid, but unspecified string.

operator=() [2/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator= (
            _CharT __c)  [inline]
```

Set value to string of length 1.

Parameters

_↩ *_c*	Source character.

Assigning to a character makes this string length 1 and (*this)[0] == *__c*.

operator=() [3/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator= (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str)  [inline]
```

Assign the value of *str* to this string.

Parameters

__str	Source string.

operator=() [4/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator= (
            const _CharT * __s)  [inline]
```

Copy contents of *__s* into this string.

Parameters

_↩ _s	Source null-terminated string.

operator=() [5/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator= (
            std::initializer_list< _CharT > __l)  [inline]
```

Set value to string constructed from initializer list.

Parameters

↩ *_↩ ↩ _↩ l*	std::initializer_list.

operator[]() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator[] (
            size_type __pos) const  [inline], [noexcept]
```

Subscript access to the data contained in the string.

Parameters

__pos	The index of the character to access.

Returns

Read-only (constant) reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out_of_range lookups are not defined. (For checked lookups see at().)
Referenced by __gnu_cxx::__versa_string< char >::back(), __gnu_cxx::__versa_string< char >::back(), __gnu_cxx::__versa_string< ch
and __gnu_cxx::__versa_string< char >::front().

operator[]() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator[] (
            size_type __pos) [inline], [noexcept]
```

Subscript access to the data contained in the string.

Parameters

__pos	The index of the character to access.

Returns

Read/write reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out_of_range lookups are not defined. (For checked lookups see at().) Unshares the string.

pop_back()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::pop_back () [inline]
```

Remove the last character.
The string must be non-empty.

push_back()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::push_back (
            _CharT __c) [inline]
```

Append a single character.

Parameters

__c	Character to append.

Referenced by __gnu_cxx::operator+(), __gnu_cxx::operator+(), and __gnu_cxx::__versa_string< char >::operator+=().

rbegin() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rbegin ()
const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

rbegin() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rbegin () [inline], [noexcept]
Returns a read/write reverse iterator that points to the last character in the string. Iteration is done in reverse element order. Unshares the string.

rend() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rend () const
[inline], [noexcept]
```
Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

rend() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rend () [inline],
[noexcept]
```
Returns a read/write reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order. Unshares the string.

replace() [1/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
            const_iterator __i1,
            const_iterator __i2,
            _InputIterator __k1,
            _InputIterator __k2) [inline]
```
Replace range of characters with range.

Parameters

__i1	Iterator referencing start of range to replace.
__i2	Iterator referencing end of range to replace.
__k1	Iterator referencing start of range to insert.
__k2	Iterator referencing end of range to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [i1,i2). In place, characters in the range [k1,k2) are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [2/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
            const_iterator __i1,
            const_iterator __i2,
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str)  [inline]
```

Replace range of characters with string.

Parameters

__i1	Iterator referencing start of range to replace.
__i2	Iterator referencing end of range to replace.
__str	String value to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [i1,i2). In place, the value of *__str* is inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [3/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
            const_iterator __i1,
            const_iterator __i2,
            const _CharT * __s)  [inline]
```

Replace range of characters with C string.

Parameters

_↩ _i1	Iterator referencing start of range to replace.
_↩ _i2	Iterator referencing end of range to replace.
_↩ _s	C string value to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [i1,i2). In place, the characters of *__s* are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [4/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
            const_iterator __i1,
            const_iterator __i2,
            const _CharT * __s,
            size_type __n)  [inline]
```

Replace range of characters with C substring.

Parameters

_i1	Iterator referencing start of range to replace.
_i2	Iterator referencing end of range to replace.
_s	C string value to insert.
_n	Number of characters from s to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [i1,i2). In place, the first *n* characters of *__s* are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [5/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
            const_iterator __i1,
            const_iterator __i2,
            size_type __n,
            _CharT __c)  [inline]
```

Replace range of characters with multiple characters.

Parameters

_↩ _i1	Iterator referencing start of range to replace.
_↩ _i2	Iterator referencing end of range to replace.
_↩ _n	Number of characters to insert.
_↩ _c	Character to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [i1,i2). In place, __*n* copies of __*c* are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [6/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
             const_iterator __i1,
             const_iterator __i2,
             std::initializer_list< _CharT > __l)  [inline]
```

Replace range of characters with initializer_list.

Parameters

_↩ _i1	Iterator referencing start of range to replace.
_↩ _i2	Iterator referencing end of range to replace.
_↩ _l	The initializer_list of characters to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [i1,i2). In place, characters in the range [k1,k2) are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [7/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
            size_type __pos,
            size_type __n,
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str)  [inline]
```

Replace characters with value from another string.

Parameters

__pos	Index of first character to replace.
__n	Number of characters to be replaced.
__str	String to insert.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *__pos* is beyond the end of this string.
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [pos,pos+n) from this string. In place, the value of *__str* is inserted. If *__pos* is beyond end of string, out_of_range is thrown. If the length of the result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.
Referenced by __gnu_cxx::__versa_string< char >::append(), __gnu_cxx::__versa_string< char >::assign(), __gnu_cxx::__versa_string< char >::insert(), __gnu_cxx::__versa_string< char >::insert(), __gnu_cxx::__versa_string< char >::insert(__gnu_cxx::__versa_string< char >::insert(), __gnu_cxx::__versa_string< char >::insert(), __gnu_cxx::__versa_string< char >::insert(__gnu_cxx::__versa_string< char >::replace(), __gnu_cxx::__versa_string< char >::replace(), __gnu_cxx::__versa_string< char >::rep __gnu_cxx::__versa_string< char >::replace(), __gnu_cxx::__versa_string< char >::replace(), __gnu_cxx::__versa_string< char >::rep and __gnu_cxx::__versa_string< char >::replace().

replace() [8/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
            size_type __pos,
            size_type __n1,
            const _CharT * __s)  [inline]
```

Replace characters with value of a C string.

Parameters

__pos	Index of first character to replace.
__n1	Number of characters to be replaced.
__s	C string to insert.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *__pos* > size().
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [pos,pos + n1) from this string. In place, the characters of *__s* are inserted. If *pos* is beyond end of string, out_of_range is thrown. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [9/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
            size_type __pos,
            size_type __n1,
            const _CharT * __s,
            size_type __n2)  [inline]
```

Replace characters with value of a C substring.

Parameters

__pos	Index of first character to replace.
__n1	Number of characters to be replaced.
__s	C string to insert.
__n2	Number of characters from *__s* to use.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *__pos1* > size().
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [pos,pos + n1) from this string. In place, the first *__n2* characters of *__s* are inserted, or all of *__s* if *__n2* is too large. If *__pos* is beyond end of string, out_of_range is thrown. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [10/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
            size_type __pos,
            size_type __n1,
            size_type __n2,
            _CharT __c)  [inline]
```

Replace characters with multiple characters.

Parameters

__pos	Index of first character to replace.
__n1	Number of characters to be replaced.
__n2	Number of characters to insert.
__c	Character to insert.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *__pos* > size().
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [pos,pos + n1) from this string. In place, *__n2* copies of *__c* are inserted. If *__pos* is beyond end of string, out_of_range is thrown. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [11/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
            size_type __pos1,
            size_type __n1,
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
            size_type __pos2,
            size_type __n2)  [inline]
```

Replace characters with value from another string.

Parameters

__pos1	Index of first character to replace.
__n1	Number of characters to be replaced.
__str	String to insert.
__pos2	Index of first character of str to use.
__n2	Number of characters from str to use.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *__pos1* > size() or *__pos2* > str.size().
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [pos1,pos1 + n) from this string. In place, the value of *__str* is inserted. If *__pos* is beyond end of string, out_of_range is thrown. If the length of the result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

reserve()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::reserve (
            size_type __res_arg = 0)  [inline]
```

Attempt to preallocate enough memory for specified number of characters.

Parameters

__res_arg	Number of characters required.

Exceptions

std::length_error	If *__res_arg* exceeds `max_size()`.

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than max_size(), length_error is thrown.
The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.
Referenced by __gnu_cxx::operator+(), __gnu_cxx::operator+(), __gnu_cxx::operator+(), and __gnu_cxx::__versa_string< char >::shrink

resize() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::resize (
            size_type __n)  [inline]
```

Resizes the string to the specified number of characters.

Parameters

__n	Number of characters the string should contain.

This function will resize the string to the specified length. If the new size is smaller than the string's current size the string is truncated, otherwise the string is extended and new characters are default-constructed. For basic types such as char, this means setting them to 0.

resize() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::resize (
            size_type __n,
            _CharT __c)
```

Resizes the string to the specified number of characters.

Parameters

__n	Number of characters the string should contain.

_↩ _c	Character to fill any new elements.

This function will resize the string to the specified number of characters. If the number is smaller than the string's current size the string is truncated, otherwise the string is extended and new elements are set to __c.
References append(), and size().
Referenced by __gnu_cxx::__versa_string< char >::resize().

rfind() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↩
_Traits, _Alloc, _Base >::rfind (
            _CharT __c,
            size_type __pos = npos) const  [noexcept]
```

Find last position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from __pos, searches backward for __c within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and size().

rfind() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rfind (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
            size_type __pos = npos) const  [inline], [noexcept]
```

Find last position of a string.

Parameters

__str	String to locate.
__pos	Index of character to search back from (default end).

Returns

Index of start of last occurrence.

Starting from __pos, searches backward for value of __str within this string. If found, returns the index where it begins. If not found, returns npos.
Referenced by __gnu_cxx::__versa_string< char >::find_last_of(), __gnu_cxx::__versa_string< char >::rfind(), and __gnu_cxx::__versa_string< char >::rfind().

rfind() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↩
_Traits, _Alloc, _Base >::rfind (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const
```

Find last position of a C substring.

Parameters

__s	C string to locate.
__pos	Index of character to search back from.
__n	Number of characters from s to search for.

Returns

Index of start of last occurrence.

Starting from *__pos*, searches backward for the first *__n* characters in *__s* within this string. If found, returns the index where it begins. If not found, returns npos.
References std::min(), npos, and size().

rfind() [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rfind (
            const _CharT * __s,
            size_type __pos = npos) const  [inline]
```

Find last position of a C string.

Parameters

__s	C string to locate.
__pos	Index of character to start search at (default end).

Returns

Index of start of last occurrence.

Starting from *__pos*, searches backward for the value of *__s* within this string. If found, returns the index where it begins. If not found, returns npos.

shrink_to_fit()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::shrink_to_fit ()  [inline],
[noexcept]
```

A non-binding request to reduce capacity() to size().

size()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size () const  [inline],
[noexcept]
```

Returns the number of characters in the string, not including any null-termination.

Referenced by __gnu_cxx::__versa_string< char >::append(), __gnu_cxx::__versa_string< char >::append(), __gnu_cxx::__versa_string< char >::assign(), __gnu_cxx::__versa_string< char >::assign(), __gnu_cxx::__versa_string< char >::assig __gnu_cxx::__versa_string< char >::assign(), __gnu_cxx::__versa_string< char >::at(), __gnu_cxx::__versa_string< char >::at(), __gnu_cxx::__versa_string< char >::back(), __gnu_cxx::__versa_string< char >::back(), __gnu_cxx::__versa_string< char >::cend(), compare(), compare(), __gnu_cxx::__versa_string< char >::compare(), __gnu_cxx::__versa_string< char >::empty(), __gnu_cxx::__versa_string< char >::end(), __gnu_cxx::__versa_string< char >::end(), find(), find(), __gnu_cxx::__versa_string< char > find_first_not_of(), find_first_not_of(), __gnu_cxx::__versa_string< char >::find_first_not_of(), find_first_of(), __gnu_cxx::__versa_string< find_last_not_of(), find_last_not_of(), find_last_of(), __gnu_cxx::__versa_string< char >::find_last_of(), __gnu_cxx::__versa_string< cha __gnu_cxx::operator+(), __gnu_cxx::operator+(), __gnu_cxx::operator+(), __gnu_cxx::operator+(), __gnu_cxx::operator+(), __gnu_cxx::operator==(), __gnu_cxx::operator==(), __gnu_cxx::__versa_string< char >::operator[](), __gnu_cxx::__versa_string< char __gnu_cxx::__versa_string< char >::pop_back(), __gnu_cxx::__versa_string< char >::push_back(), __gnu_cxx::__versa_string< char __gnu_cxx::__versa_string< char >::replace(), resize(), rfind(), rfind(), and __gnu_cxx::__versa_string< char >::shrink_to_fit().

substr()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::substr (
            size_type __pos = 0,
            size_type __n = npos) const  [inline]
```

Get a substring.

Parameters

__pos	Index of first character (default 0).
__n	Number of characters in substring (default remainder).

Returns

The new string.

Exceptions

std::out_of_range	If pos > size().

Construct and return a new string using the *__n* characters starting at *__pos*. If the string is too short, use the remainder of the characters. If *__pos* is beyond the end of the string, out_of_range is thrown.

swap()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::swap (
            __versa_string< _CharT, _Traits, _Alloc, _Base > & __s)  [inline], [noexcept]
```

Swap contents with another string.

Parameters

_↩ _s	String to swap with.

Exchanges the contents of this string with that of __*s* in constant time.
Referenced by __gnu_cxx::__versa_string< char >::assign(), __gnu_cxx::__versa_string< char >::operator=(), and __gnu_cxx::swap().

5.58.4 Member Data Documentation

npos

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::npos  [static]
```

Value returned by various member functions when they fail.
Referenced by find(), find(), find_first_not_of(), find_first_not_of(), find_first_of(), find_last_not_of(), find_last_of(), rfind(), and rfind().
The documentation for this class was generated from the following files:

- vstring.h
- vstring.tcc

5.59 __gnu_debug::_After_nth_from< _Iterator > Class Template Reference

```
#include <safe_sequence.h>
```

Public Member Functions

- **_After_nth_from** (const difference_type &__n, const _Iterator &__base)
- bool **operator()** (const _Iterator &__x) const

5.59.1 Detailed Description

template<typename _Iterator>
class __gnu_debug::_After_nth_from< _Iterator >

A function object that returns true when the given random access iterator is at least `n` steps away from the given iterator.
The documentation for this class was generated from the following file:

- safe_sequence.h

5.60 std::_Base_bitset< _Nw > Struct Template Reference

```
#include <bitset>
```

Public Types

- typedef unsigned long **_WordT**

Public Member Functions

- constexpr **_Base_bitset** (unsigned long long __val) noexcept
- template<size_t _Nb>
  constexpr bool **_M_are_all** () const noexcept

- constexpr void **_M_do_and** (const _Base_bitset< _Nw > &__x) noexcept
- constexpr size_t **_M_do_count** () const noexcept
- constexpr size_t **_M_do_find_first** (size_t) const noexcept
- constexpr size_t **_M_do_find_next** (size_t, size_t) const noexcept
- constexpr void **_M_do_flip** () noexcept
- constexpr void **_M_do_left_shift** (size_t __shift) noexcept
- constexpr void **_M_do_or** (const _Base_bitset< _Nw > &__x) noexcept
- constexpr void **_M_do_reset** () noexcept
- constexpr void **_M_do_right_shift** (size_t __shift) noexcept
- constexpr void **_M_do_set** () noexcept
- constexpr unsigned long long **_M_do_to_ullong** () const
- constexpr unsigned long **_M_do_to_ulong** () const
- constexpr void **_M_do_xor** (const _Base_bitset< _Nw > &__x) noexcept
- constexpr const _WordT * **_M_getdata** () const noexcept
- constexpr _WordT **_M_getword** (size_t __pos) const noexcept
- constexpr _WordT & **_M_getword** (size_t __pos) noexcept
- constexpr _WordT **_M_hiword** () const noexcept
- constexpr _WordT & **_M_hiword** () noexcept
- constexpr bool **_M_is_any** () const noexcept
- constexpr bool **_M_is_equal** (const _Base_bitset< _Nw > &__x) const noexcept

Static Public Member Functions

- static constexpr _WordT **_S_maskbit** (size_t __pos) noexcept
- static constexpr size_t **_S_whichbit** (size_t __pos) noexcept
- static constexpr size_t **_S_whichbyte** (size_t __pos) noexcept
- static constexpr size_t **_S_whichword** (size_t __pos) noexcept

Public Attributes

- _WordT _M_w [_Nw]

5.60.1 Detailed Description

template<size_t _Nw>
struct std::_Base_bitset< _Nw >

Base class, general case. It is a class invariant that _Nw will be nonnegative.
See documentation for bitset.

5.60.2 Member Data Documentation

_M_w

```
template<size_t _Nw>
_WordT std::_Base_bitset< _Nw >::_M_w[_Nw]
```

0 is the least significant word.
The documentation for this struct was generated from the following file:

- bitset

5.61 std::_Base_bitset< 0 > Struct Reference

```
#include <bitset>
```

Public Types

- typedef unsigned long **_WordT**
- typedef unsigned long **_WordT**

Public Member Functions

- constexpr **_Base_bitset** (unsigned long long __val) noexcept
- constexpr **_Base_bitset** (unsigned long long) noexcept
- constexpr bool **_M_are_all** () const noexcept
- template<size_t _Nb>
  constexpr bool **_M_are_all** () const noexcept
- constexpr void **_M_do_and** (const _Base_bitset< 0 > &) noexcept
- constexpr void **_M_do_and** (const _Base_bitset< _Nw > &__x) noexcept
- constexpr size_t **_M_do_count** () const noexcept
- constexpr size_t **_M_do_count** () const noexcept
- constexpr size_t **_M_do_find_first** (size_t) const noexcept
- constexpr size_t **_M_do_find_first** (size_t) const noexcept
- constexpr size_t **_M_do_find_next** (size_t, size_t) const noexcept
- constexpr size_t **_M_do_find_next** (size_t, size_t) const noexcept
- constexpr void **_M_do_flip** () noexcept
- constexpr void **_M_do_flip** () noexcept
- constexpr void **_M_do_left_shift** (size_t __shift) noexcept
- constexpr void **_M_do_left_shift** (size_t) noexcept
- constexpr void **_M_do_or** (const _Base_bitset< 0 > &) noexcept
- constexpr void **_M_do_or** (const _Base_bitset< _Nw > &__x) noexcept
- constexpr void **_M_do_reset** () noexcept
- constexpr void **_M_do_reset** () noexcept
- constexpr void **_M_do_right_shift** (size_t __shift) noexcept
- constexpr void **_M_do_right_shift** (size_t) noexcept
- constexpr void **_M_do_set** () noexcept
- constexpr void **_M_do_set** () noexcept
- constexpr unsigned long long **_M_do_to_ullong** () const
- constexpr unsigned long long **_M_do_to_ullong** () const noexcept
- constexpr unsigned long **_M_do_to_ulong** () const
- constexpr unsigned long **_M_do_to_ulong** () const noexcept
- constexpr void **_M_do_xor** (const _Base_bitset< 0 > &) noexcept
- constexpr void **_M_do_xor** (const _Base_bitset< _Nw > &__x) noexcept
- constexpr const _WordT * **_M_getdata** () const noexcept
- constexpr _WordT **_M_getword** (size_t __pos) const noexcept
- constexpr _WordT & **_M_getword** (size_t __pos) noexcept
- constexpr _WordT **_M_getword** (size_t) const noexcept
- _WordT & **_M_getword** (size_t) noexcept
- constexpr _WordT **_M_hiword** () const noexcept
- constexpr _WordT **_M_hiword** () const noexcept
- constexpr _WordT & **_M_hiword** () noexcept
- constexpr bool **_M_is_any** () const noexcept
- constexpr bool **_M_is_any** () const noexcept
- constexpr bool **_M_is_equal** (const _Base_bitset< 0 > &) const noexcept
- constexpr bool **_M_is_equal** (const _Base_bitset< _Nw > &__x) const noexcept

Static Public Member Functions

- static constexpr _WordT **_S_maskbit** (size_t __pos) noexcept
- static constexpr _WordT **_S_maskbit** (size_t __pos) noexcept
- static constexpr size_t **_S_whichbit** (size_t __pos) noexcept
- static constexpr size_t **_S_whichbit** (size_t __pos) noexcept
- static constexpr size_t **_S_whichbyte** (size_t __pos) noexcept
- static constexpr size_t **_S_whichbyte** (size_t __pos) noexcept
- static constexpr size_t **_S_whichword** (size_t __pos) noexcept
- static constexpr size_t **_S_whichword** (size_t __pos) noexcept

Public Attributes

- _WordT _M_w [_Nw]

5.61.1 Detailed Description

Base class, specialization for no storage (zero-length bitset).
See documentation for bitset.

5.61.2 Member Data Documentation

_M_w

`_WordT std::_Base_bitset< _Nw >::_M_w[_Nw]`
0 is the least significant word.
The documentation for this struct was generated from the following file:

- bitset

5.62 std::_Base_bitset< 1 > Struct Reference

`#include <bitset>`

Public Types

- typedef unsigned long **_WordT**
- typedef unsigned long **_WordT**

Public Member Functions

- constexpr **_Base_bitset** (unsigned long long __val) noexcept
- constexpr **_Base_bitset** (unsigned long long __val) noexcept
- constexpr bool **_M_are_all** () const noexcept
- template<size_t _Nb>
  constexpr bool **_M_are_all** () const noexcept
- constexpr void **_M_do_and** (const _Base_bitset< 1 > &__x) noexcept
- constexpr void **_M_do_and** (const _Base_bitset< _Nw > &__x) noexcept
- constexpr size_t **_M_do_count** () const noexcept
- constexpr size_t **_M_do_count** () const noexcept
- constexpr size_t **_M_do_find_first** (size_t __not_found) const noexcept
- constexpr size_t **_M_do_find_first** (size_t) const noexcept
- constexpr size_t **_M_do_find_next** (size_t __prev, size_t __not_found) const noexcept
- constexpr size_t **_M_do_find_next** (size_t, size_t) const noexcept
- constexpr void **_M_do_flip** () noexcept

- constexpr void **_M_do_flip** () noexcept
- constexpr void **_M_do_left_shift** (size_t __shift) noexcept
- constexpr void **_M_do_left_shift** (size_t __shift) noexcept
- constexpr void **_M_do_or** (const _Base_bitset< 1 > &__x) noexcept
- constexpr void **_M_do_or** (const _Base_bitset< _Nw > &__x) noexcept
- constexpr void **_M_do_reset** () noexcept
- constexpr void **_M_do_reset** () noexcept
- constexpr void **_M_do_right_shift** (size_t __shift) noexcept
- constexpr void **_M_do_right_shift** (size_t __shift) noexcept
- constexpr void **_M_do_set** () noexcept
- constexpr void **_M_do_set** () noexcept
- constexpr unsigned long long **_M_do_to_ullong** () const
- constexpr unsigned long long **_M_do_to_ullong** () const noexcept
- constexpr unsigned long **_M_do_to_ulong** () const
- constexpr unsigned long **_M_do_to_ulong** () const noexcept
- constexpr void **_M_do_xor** (const _Base_bitset< 1 > &__x) noexcept
- constexpr void **_M_do_xor** (const _Base_bitset< _Nw > &__x) noexcept
- constexpr const _WordT * **_M_getdata** () const noexcept
- constexpr const _WordT * **_M_getdata** () const noexcept
- constexpr _WordT **_M_getword** (size_t __pos) const noexcept
- constexpr _WordT & **_M_getword** (size_t __pos) noexcept
- constexpr _WordT **_M_getword** (size_t) const noexcept
- constexpr _WordT & **_M_getword** (size_t) noexcept
- constexpr _WordT **_M_hiword** () const noexcept
- constexpr _WordT **_M_hiword** () const noexcept
- constexpr _WordT & **_M_hiword** () noexcept
- constexpr _WordT & **_M_hiword** () noexcept
- constexpr bool **_M_is_any** () const noexcept
- constexpr bool **_M_is_any** () const noexcept
- constexpr bool **_M_is_equal** (const _Base_bitset< 1 > &__x) const noexcept
- constexpr bool **_M_is_equal** (const _Base_bitset< _Nw > &__x) const noexcept

Static Public Member Functions

- static constexpr _WordT **_S_maskbit** (size_t __pos) noexcept
- static constexpr _WordT **_S_maskbit** (size_t __pos) noexcept
- static constexpr size_t **_S_whichbit** (size_t __pos) noexcept
- static constexpr size_t **_S_whichbit** (size_t __pos) noexcept
- static constexpr size_t **_S_whichbyte** (size_t __pos) noexcept
- static constexpr size_t **_S_whichbyte** (size_t __pos) noexcept
- static constexpr size_t **_S_whichword** (size_t __pos) noexcept
- static constexpr size_t **_S_whichword** (size_t __pos) noexcept

Public Attributes

- _WordT _M_w [_Nw]
- _WordT **_M_w**

5.62.1 Detailed Description

Base class, specialization for a single word.
See documentation for bitset.

5.62.2 Member Data Documentation

_M_w

_WordT std::_Base_bitset< _Nw >::_M_w[_Nw]
0 is the least significant word.
The documentation for this struct was generated from the following file:

- bitset

5.63 __gnu_debug::_BeforeBeginHelper< _Sequence > Struct Template Reference

#include <safe_iterator.h>

Static Public Member Functions

- template<typename _Iterator, typename _Category>
  static bool **_S_Is** (const _Safe_iterator< _Iterator, _Sequence, _Category > &)
- template<typename _Iterator, typename _Category>
  static bool **_S_Is_Beginnest** (const _Safe_iterator< _Iterator, _Sequence, _Category > &__it)

5.63.1 Detailed Description

template<typename _Sequence>
struct __gnu_debug::_BeforeBeginHelper< _Sequence >

Helper struct to deal with sequence offering a before_begin iterator.
The documentation for this struct was generated from the following file:

- safe_iterator.h

5.64 std::_Bind< _Signature > Class Template Reference

5.64.1 Detailed Description

template<typename _Signature>
class std::_Bind< _Signature >

Type of the function object returned from bind().
The documentation for this class was generated from the following file:

- functional

5.65 std::_Bind_result< _Result, _Signature > Class Template Reference

5.65.1 Detailed Description

template<typename _Result, typename _Signature>
class std::_Bind_result< _Result, _Signature >

Type of the function object returned from bind<R>().
The documentation for this class was generated from the following file:

- functional

5.66 __gnu_cxx::__detail::_Bitmap_counter< _Tp > Class Template Reference

#include <bitmap_allocator.h>

Public Member Functions

- **_Bitmap_counter** (_BPVector &Rvbp, long __index=-1)
- pointer **_M_base** () const throw ()
- bool **_M_finished** () const throw ()
- std::size_t * **_M_get** () const throw ()
- _Index_type **_M_offset** () const throw ()
- void **_M_reset** (long __index=-1) throw ()
- void **_M_set_internal_bitmap** (std::size_t *__new_internal_marker) throw ()
- _Index_type **_M_where** () const throw ()
- _Bitmap_counter & **operator++** () throw ()

5.66.1 Detailed Description

template<typename _Tp>
class __gnu_cxx::__detail::_Bitmap_counter< _Tp >

The bitmap counter which acts as the bitmap manipulator, and manages the bit-manipulation functions and the searching and identification functions on the bit-map.
The documentation for this class was generated from the following file:

- bitmap_allocator.h

5.67 std::__detail::_BracketMatcher< _TraitsT, __icase, __collate > Struct Template Reference

`#include <regex_compiler.h>`

Public Types

- typedef _TraitsT::char_class_type **_CharClassT**
- typedef _TransT::_CharT **_CharT**
- typedef _TraitsT::string_type **_StringT**
- typedef _TransT::_StrTransT **_StrTransT**
- typedef _RegexTranslator< _TraitsT, __icase, __collate > **_TransT**

Public Member Functions

- **_BracketMatcher** (bool __is_non_matching, const _TraitsT &__traits)
- void **_M_add_char** (_CharT __c)
- void **_M_add_character_class** (const _StringT &__s, bool __neg)
- _StringT **_M_add_collate_element** (const _StringT &__s)
- void **_M_add_equivalence_class** (const _StringT &__s)
- void **_M_make_range** (_CharT __l, _CharT __r)
- void **_M_ready** ()
- bool **operator()** (_CharT __ch) const

5.67.1 Detailed Description

template<typename _TraitsT, bool __icase, bool __collate>
struct std::__detail::_BracketMatcher< _TraitsT, __icase, __collate >

Matches a character range (bracket expression)
The documentation for this struct was generated from the following files:

- regex_compiler.h
- regex_compiler.tcc

5.68 __gnu_cxx::_Caster< _ToType > Struct Template Reference

`#include <cast.h>`

Public Types

- typedef _ToType::element_type * **type**

5.68.1 Detailed Description

template<typename _ToType>
struct __gnu_cxx::_Caster< _ToType >

These functions are here to allow containers to support non standard pointer types. For normal pointers, these resolve to the use of the standard cast operation. For other types the functions will perform the appropriate cast to/from the custom pointer class so long as that class meets the following conditions: 1) has a typedef element_type which names tehe type it points to. 2) has a get() const method which returns element_type*. 3) has a constructor which can take one element_type* argument. This type supports the semantics of the pointer cast operators (below.)
The documentation for this struct was generated from the following file:

- cast.h

5.69 __gnu_cxx::_Char_types< _CharT > Struct Template Reference

`#include <char_traits.h>`

Public Types

- typedef unsigned long **int_type**
- typedef std::streamoff **off_type**
- typedef std::streampos **pos_type**
- typedef std::mbstate_t **state_type**

5.69.1 Detailed Description

template<typename _CharT>
struct __gnu_cxx::_Char_types< _CharT >

Mapping from character type to associated types.

Note

This is an implementation class for the generic version of char_traits. It defines int_type, off_type, pos_type, and state_type. By default these are unsigned long, streamoff, streampos, and mbstate_t. Users who need a different set of types, but who don't need to change the definitions of any function defined in char_traits, can specialize __gnu_cxx::_Char_types while leaving __gnu_cxx::char_traits alone.

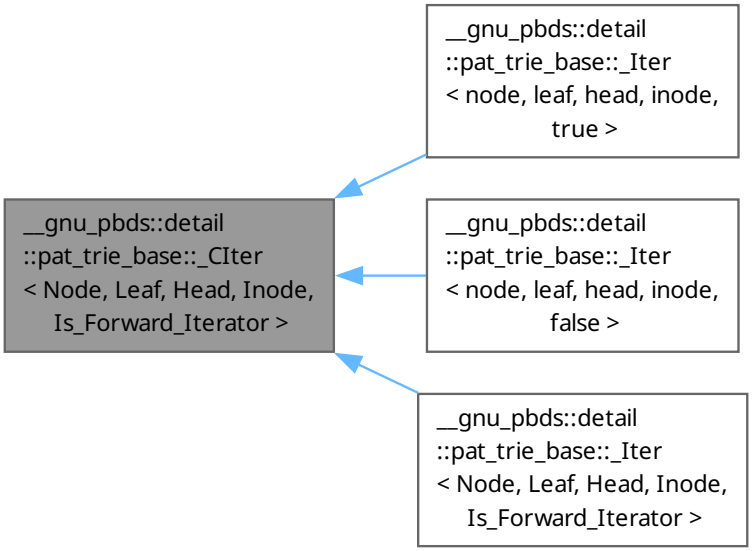
The documentation for this struct was generated from the following file:

- char_traits.h

5.70 __gnu_pbds::detail::pat_trie_base::_CIter< Node, Leaf, Head, Inode, Is_Forward_Iterator > Class Template Reference

`#include <pat_trie_base.hpp>`

Inheritance diagram for __gnu_pbds::detail::pat_trie_base::_CIter< Node, Leaf, Head, Inode, Is_Forward_Iterator >:



Public Types

- typedef allocator_type **_Alloc**
- typedef Node::allocator_type **allocator_type**
- typedef type_traits::const_pointer **const_pointer**
- typedef type_traits::const_reference **const_reference**
- typedef allocator_type::difference_type **difference_type**
- typedef rebind_traits< _Alloc, Head >::pointer **head_pointer**
- typedef Inode::iterator **inode_iterator**
- typedef rebind_traits< _Alloc, Inode >::pointer **inode_pointer**
- typedef std::bidirectional_iterator_tag **iterator_category**
- typedef rebind_traits< _Alloc, Leaf >::const_pointer **leaf_const_pointer**
- typedef rebind_traits< _Alloc, Leaf >::pointer **leaf_pointer**
- typedef rebind_traits< _Alloc, Node >::pointer **node_pointer**
- typedef type_traits::pointer **pointer**
- typedef type_traits::reference **reference**
- typedef Node::type_traits **type_traits**
- typedef type_traits::value_type **value_type**

Public Member Functions

- **_CIter** (const _CIter< Node, Leaf, Head, Inode, !Is_Forward_Iterator > &other)
- **_CIter** (node_pointer p_nd=0)
- bool **operator!=** (const _CIter &other) const
- bool **operator!=** (const _CIter< Node, Leaf, Head, Inode, !Is_Forward_Iterator > &other) const
- const_reference **operator*** () const

- _CIter & **operator++** ()
- _CIter **operator++** (int)
- _CIter & **operator--** ()
- _CIter **operator--** (int)
- const_pointer **operator->** () const
- _CIter & **operator=** (const _CIter &other)
- _CIter & **operator=** (const _CIter< Node, Leaf, Head, Inode, !Is_Forward_Iterator > &other)
- bool **operator==** (const _CIter &other) const
- bool **operator==** (const _CIter< Node, Leaf, Head, Inode, !Is_Forward_Iterator > &other) const

Public Attributes

- node_pointer **m_p_nd**

Protected Member Functions

- void **dec** (false_type)
- void **dec** (true_type)
- void **inc** (false_type)
- void **inc** (true_type)

Static Protected Member Functions

- static node_pointer **get_larger_sibling** (node_pointer p_nd)
- static node_pointer **get_smaller_sibling** (node_pointer p_nd)
- static leaf_pointer **leftmost_descendant** (node_pointer p_nd)
- static leaf_pointer **rightmost_descendant** (node_pointer p_nd)

5.70.1 Detailed Description

template<typename Node, typename Leaf, typename Head, typename Inode, bool Is_Forward_Iterator>
class __gnu_pbds::detail::pat_trie_base::_CIter< Node, Leaf, Head, Inode, Is_Forward_Iterator >

Const iterator.
The documentation for this class was generated from the following file:

- pat_trie_base.hpp

5.71 std::__detail::_Compiler< _TraitsT > Class Template Reference

```
#include <regex_compiler.h>
```

Public Types

- typedef _TraitsT::char_type **_CharT**
- typedef regex_constants::syntax_option_type **_FlagT**
- typedef _NFA< _TraitsT > **_RegexT**

Public Member Functions

- **_Compiler** (const _CharT *__b, const _CharT *__e, const typename _TraitsT::locale_type &__traits, _FlagT __flags)
- shared_ptr< const _RegexT > **_M_get_nfa** () noexcept

5.71.1 Detailed Description

template<typename _TraitsT>
class std::__detail::_Compiler< _TraitsT >

Builds an NFA from an input iterator range.
The _TraitsT type should fulfill requirements [28.3].
The documentation for this class was generated from the following files:

- regex_compiler.h
- regex_compiler.tcc

5.72 std::__parallel::_CRandNumber< _MustBeInt > Struct Template Reference

`#include <algo.h>`

Public Member Functions

- int **operator()** (int __limit)

5.72.1 Detailed Description

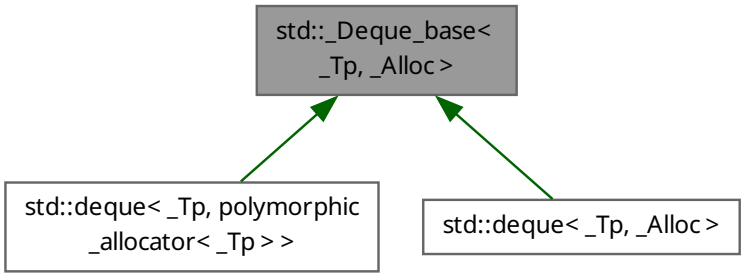
template<typename _MustBeInt = int>
struct std::__parallel::_CRandNumber< _MustBeInt >

Functor wrapper for std::rand().
The documentation for this struct was generated from the following file:

- algo.h

5.73 std::_Deque_base< _Tp, _Alloc > Class Template Reference

`#include <stl_deque.h>`
Inheritance diagram for std::_Deque_base< _Tp, _Alloc >:



Protected Types

- enum { **_S_initial_map_size** }
- typedef __gnu_cxx::__alloc_traits< _Tp_alloc_type > **_Alloc_traits**
- typedef __gnu_cxx::__alloc_traits< _Map_alloc_type > **_Map_alloc_traits**

- typedef _Alloc_traits::template rebind< _Ptr >::other **_Map_alloc_type**
- typedef iterator::_Map_pointer **_Map_pointer**
- typedef _Alloc_traits::pointer **_Ptr**
- typedef _Alloc_traits::const_pointer **_Ptr_const**
- typedef __gnu_cxx::__alloc_traits< _Alloc >::template rebind< _Tp >::other **_Tp_alloc_type**
- typedef _Alloc **allocator_type**
- typedef _Deque_iterator< _Tp, const _Tp &, _Ptr_const > **const_iterator**
- typedef _Deque_iterator< _Tp, _Tp &, _Ptr > **iterator**

Protected Member Functions

- **_Deque_base** (_Deque_base &&__x)
- **_Deque_base** (_Deque_base &&__x, const allocator_type &__a)
- **_Deque_base** (_Deque_base &&__x, const allocator_type &__a, size_t __n)
- **_Deque_base** (const allocator_type &__a)
- **_Deque_base** (const allocator_type &__a, size_t __num_elements)
- **_Deque_base** (size_t __num_elements)
- _Map_pointer **_M_allocate_map** (size_t __n)
- _Ptr **_M_allocate_node** ()
- void **_M_create_nodes** (_Map_pointer __nstart, _Map_pointer __nfinish)
- void **_M_deallocate_map** (_Map_pointer __p, size_t __n) noexcept
- void **_M_deallocate_node** (_Ptr __p) noexcept
- void **_M_destroy_nodes** (_Map_pointer __nstart, _Map_pointer __nfinish) noexcept
- _Map_alloc_type **_M_get_map_allocator** () const noexcept
- const _Tp_alloc_type & **_M_get_Tp_allocator** () const noexcept
- _Tp_alloc_type & **_M_get_Tp_allocator** () noexcept
- void _M_initialize_map (size_t)
- allocator_type **get_allocator** () const noexcept

Protected Attributes

- _Deque_impl **_M_impl**

5.73.1 Detailed Description

template<typename _Tp, typename _Alloc>
class std::_Deque_base< _Tp, _Alloc >

Deque base class. This class provides the unified face for deque's allocation. This class's constructor and destructor allocate and deallocate (but do not initialize) storage. This makes exception safety easier.
Nothing in this class ever constructs or destroys an actual Tp element. (Deque handles that itself.) Only/All memory management is performed here.

5.73.2 Member Function Documentation

_M_initialize_map()

```
template<typename _Tp, typename _Alloc>
void std::_Deque_base< _Tp, _Alloc >::_M_initialize_map (
            size_t __num_elements)  [protected]
```

Layout storage.

Parameters

__num_elements	The count of T's for which to allocate space at first.

Returns

Nothing.

The initial underlying memory layout is a bit complicated...
References std::max().
The documentation for this class was generated from the following file:

- stl_deque.h

5.74 std::_Deque_iterator< _Tp, _Ref, _Ptr > Struct Template Reference

`#include <stl_deque.h>`

Public Types

- typedef __ptr_rebind< _Ptr, _Tp > **_Elt_pointer**
- typedef __ptr_rebind< _Ptr, _Elt_pointer > **_Map_pointer**
- typedef _Deque_iterator **_Self**
- typedef __iter< const _Tp > **const_iterator**
- typedef ptrdiff_t **difference_type**
- typedef __iter< _Tp > **iterator**
- typedef std::random_access_iterator_tag **iterator_category**
- typedef _Ptr **pointer**
- typedef _Ref **reference**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **_Deque_iterator** (_Elt_pointer __x, _Map_pointer __y) noexcept
- **_Deque_iterator** (const _Deque_iterator &__x) noexcept
- template<typename _Iter, typename = _Require<is_same<_Self, const_iterator>, is_same<_Iter, iterator>>> **_Deque_iterator** (const _Iter &__x) noexcept
- iterator **_M_const_cast** () const noexcept
- void _M_set_node (_Map_pointer __new_node) noexcept
- reference **operator*** () const noexcept
- _Self & **operator++** () noexcept
- _Self **operator++** (int) noexcept
- _Self & **operator+=** (difference_type __n) noexcept
- _Self & **operator--** () noexcept
- _Self **operator--** (int) noexcept
- _Self & **operator-=** (difference_type __n) noexcept
- pointer **operator->** () const noexcept
- _Deque_iterator & **operator=** (const _Deque_iterator &)=default
- reference **operator[ ]** (difference_type __n) const noexcept

Static Public Member Functions

- static size_t **_S_buffer_size** () noexcept

Public Attributes

- _Elt_pointer **_M_cur**
- _Elt_pointer **_M_first**
- _Elt_pointer **_M_last**
- _Map_pointer **_M_node**

Friends

- _Self **operator+** (const _Self &__x, difference_type __n) noexcept
- _Self **operator+** (difference_type __n, const _Self &__x) noexcept
- template<typename _RefR, typename _PtrR>
  difference_type **operator-** (const _Self &__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y) noexcept
- difference_type **operator-** (const _Self &__x, const _Self &__y) noexcept
- _Self **operator-** (const _Self &__x, difference_type __n) noexcept
- strong_ordering **operator<=>** (const _Self &__x, const _Self &__y) noexcept
- template<typename _RefR, typename _PtrR>
  bool **operator==** (const _Self &__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y) noexcept
- bool **operator==** (const _Self &__x, const _Self &__y) noexcept

5.74.1 Detailed Description

template<typename _Tp, typename _Ref, typename _Ptr>
struct std::_Deque_iterator< _Tp, _Ref, _Ptr >

A deque::iterator.
Quite a bit of intelligence here. Much of the functionality of deque is actually passed off to this class. A deque holds two of these internally, marking its valid range. Access to elements is done as offsets of either of those two, relying on operator overloading in this class.
All the functions are op overloads except for _M_set_node.

5.74.2 Member Function Documentation

_M_set_node()

```
template<typename _Tp, typename _Ref, typename _Ptr>
void std::_Deque_iterator< _Tp, _Ref, _Ptr >::_M_set_node (
            _Map_pointer __new_node)  [inline], [noexcept]
```

Prepares to traverse new_node. Sets everything except _M_cur, which should therefore be set by the caller immediately afterwards, based on _M_first and _M_last.
The documentation for this struct was generated from the following files:

- stl_algobase.h
- stl_deque.h

5.75 __gnu_parallel::_DRandomShufflingGlobalData< _RAIter > Struct Template Reference

```
#include <random_shuffle.h>
```

Public Types

- typedef _TraitsType::difference_type **_DifferenceType**
- typedef std::iterator_traits< _RAIter > **_TraitsType**
- typedef _TraitsType::value_type **_ValueType**

Public Member Functions

- _DRandomShufflingGlobalData (_RAIter &__source)

Public Attributes

- _ThreadIndex * _M_bin_proc
- _DifferenceType ** _M_dist
- int _M_num_bins
- int _M_num_bits
- _RAIter & _M_source
- _DifferenceType * _M_starts
- _ValueType ** _M_temporaries

5.75.1 Detailed Description

template<typename _RAIter>
struct __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >

Data known to every thread participating in __gnu_parallel::__parallel_random_shuffle().

5.75.2 Constructor & Destructor Documentation

_DRandomShufflingGlobalData()

```
template<typename _RAIter>
__gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_DRandomShufflingGlobalData (
            _RAIter & __source)  [inline]
```

Constructor.
References _M_source.

5.75.3 Member Data Documentation

_M_bin_proc

```
template<typename _RAIter>
_ThreadIndex* __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_bin_proc
```

Number of the thread that will further process the corresponding bin.
Referenced by __gnu_parallel::__parallel_random_shuffle_drs(), and __gnu_parallel::__parallel_random_shuffle_drs_pu().

_M_dist

```
template<typename _RAIter>
_DifferenceType** __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_dist
```

Two-dimensional array to hold the thread-bin distribution.
Dimensions (_M_num_threads + 1) __x (_M_num_bins + 1).
Referenced by __gnu_parallel::__parallel_random_shuffle_drs(), and __gnu_parallel::__parallel_random_shuffle_drs_pu().

_M_num_bins

```
template<typename _RAIter>
int __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_num_bins
```

Number of bins to distribute to.
Referenced by __gnu_parallel::__parallel_random_shuffle_drs(), and __gnu_parallel::__parallel_random_shuffle_drs_pu().

_M_num_bits

```
template<typename _RAIter>
int __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_num_bits
```

Number of bits needed to address the bins.
Referenced by __gnu_parallel::__parallel_random_shuffle_drs(), and __gnu_parallel::__parallel_random_shuffle_drs_pu().

_M_source

```
template<typename _RAIter>
_RAIter& __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_source
```

Begin iterator of the __source.
Referenced by _DRandomShufflingGlobalData(), and __gnu_parallel::__parallel_random_shuffle_drs_pu().

_M_starts

```
template<typename _RAIter>
_DifferenceType* __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_starts
```

Start indexes of the threads' __chunks.
Referenced by __gnu_parallel::__parallel_random_shuffle_drs(), and __gnu_parallel::__parallel_random_shuffle_drs_pu().

_M_temporaries

```
template<typename _RAIter>
_ValueType** __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_temporaries
```

Temporary arrays for each thread.
Referenced by __gnu_parallel::__parallel_random_shuffle_drs(), and __gnu_parallel::__parallel_random_shuffle_drs_pu().
The documentation for this struct was generated from the following file:

- random_shuffle.h

5.76 __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator > Struct Template Reference

```
#include <random_shuffle.h>
```

Public Attributes

- _BinIndex __bins_end
- _BinIndex _M_bins_begin
- int _M_num_threads
- _DRandomShufflingGlobalData< _RAIter > * _M_sd
- uint32_t _M_seed

5.76.1 Detailed Description

template<typename _RAIter, typename _RandomNumberGenerator>
struct __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >

Local data for a thread participating in __gnu_parallel::__parallel_random_shuffle().

5.76.2 Member Data Documentation

__bins_end

```
template<typename _RAIter, typename _RandomNumberGenerator>
_BinIndex __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::__bins_end
```

End index for bins taken care of by this thread.
Referenced by __gnu_parallel::__parallel_random_shuffle_drs().

_M_bins_begin

```
template<typename _RAIter, typename _RandomNumberGenerator>
_BinIndex __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_bins_begin
```

Begin index for bins taken care of by this thread.
Referenced by __gnu_parallel::__parallel_random_shuffle_drs(), and __gnu_parallel::__parallel_random_shuffle_drs_pu().

_M_num_threads

```
template<typename _RAIter, typename _RandomNumberGenerator>
int __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_num_threads
```

Number of threads participating in total.
Referenced by __gnu_parallel::__parallel_random_shuffle_drs(), and __gnu_parallel::__parallel_random_shuffle_drs_pu().

_M_sd

```
template<typename _RAIter, typename _RandomNumberGenerator>
_DRandomShufflingGlobalData<_RAIter>* __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumber↩
Generator >::_M_sd
```

Pointer to global data.
Referenced by __gnu_parallel::__parallel_random_shuffle_drs(), and __gnu_parallel::__parallel_random_shuffle_drs_pu().

_M_seed

```
template<typename _RAIter, typename _RandomNumberGenerator>
uint32_t __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_seed
```

Random _M_seed for this thread.
Referenced by __gnu_parallel::__parallel_random_shuffle_drs(), and __gnu_parallel::__parallel_random_shuffle_drs_pu().
The documentation for this struct was generated from the following file:

- random_shuffle.h

5.77 __gnu_parallel::_DummyReduct Struct Reference

```
#include <for_each_selectors.h>
```

Public Member Functions

- bool **operator()** (bool, bool) const

5.77.1 Detailed Description

Reduction function doing nothing.
The documentation for this struct was generated from the following file:

- for_each_selectors.h

5.78 __gnu_debug::_Equal_to< _Type > Class Template Reference

```
#include <safe_sequence.h>
```

Public Member Functions

- **_Equal_to** (const _Type &__v)
- bool **operator()** (const _Type &__x) const

5.78.1 Detailed Description

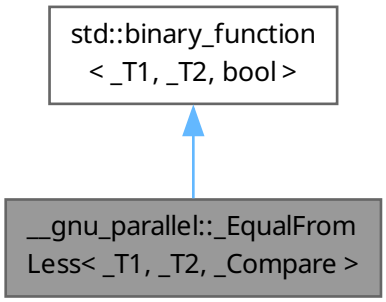
template<typename _Type>
class __gnu_debug::_Equal_to< _Type >

A simple function object that returns true if the passed-in value is equal to the stored value.
The documentation for this class was generated from the following file:

- safe_sequence.h

5.79 __gnu_parallel::_EqualFromLess< _T1, _T2, _Compare > Class Template Reference

`#include <base.h>`
Inheritance diagram for __gnu_parallel::_EqualFromLess< _T1, _T2, _Compare >:



Public Types

- typedef _T1 first_argument_type
- typedef bool result_type
- typedef _T2 second_argument_type

Public Member Functions

- **_EqualFromLess** (_Compare &__comp)
- bool **operator()** (const _T1 &__a, const _T2 &__b)

5.79.1 Detailed Description

template<typename _T1, typename _T2, typename _Compare>
class __gnu_parallel::_EqualFromLess< _T1, _T2, _Compare >

Constructs predicate for equality from strict weak ordering predicate.

5.79.2 Member Typedef Documentation

first_argument_type

`typedef _T1 std::binary_function< _T1, _T2, bool >::first_argument_type [inherited]`
`first_argument_type` is the type of the first argument

result_type

`typedef bool std::binary_function< _T1, _T2, bool >::result_type [inherited]`
`result_type` is the return type

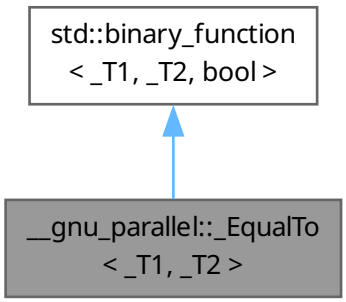
second_argument_type

`typedef _T2 std::binary_function< _T1, _T2, bool >::second_argument_type [inherited]`
`second_argument_type` is the type of the second argument
The documentation for this class was generated from the following file:

- base.h

5.80 __gnu_parallel::_EqualTo< _T1, _T2 > Struct Template Reference

`#include <base.h>`
Inheritance diagram for __gnu_parallel::_EqualTo< _T1, _T2 >:



Public Types

- typedef _T1 first_argument_type
- typedef bool result_type
- typedef _T2 second_argument_type

Public Member Functions

- bool **operator()** (const _T1 &__t1, const _T2 &__t2) const

5.80.1 Detailed Description

template<typename _T1, typename _T2>
struct __gnu_parallel::_EqualTo< _T1, _T2 >

Similar to std::equal_to, but allows two different types.

5.80.2 Member Typedef Documentation

first_argument_type

```
typedef _T1 std::binary_function< _T1, _T2, bool >::first_argument_type  [inherited]
```

`first_argument_type` is the type of the first argument

result_type

```
typedef bool std::binary_function< _T1, _T2, bool >::result_type  [inherited]
```

`result_type` is the return type

second_argument_type

```
typedef _T2 std::binary_function< _T1, _T2, bool >::second_argument_type  [inherited]
```

`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- base.h

5.81 std::__detail::_Executor< _BiIter, _Alloc, _TraitsT, __dfs_mode > Class Template Reference

`#include <regex_executor.h>`

Public Types

- typedef iterator_traits< _BiIter >::value_type **_CharT**
- typedef _TraitsT::char_class_type **_ClassT**
- typedef regex_constants::match_flag_type **_FlagT**
- typedef _NFA< _TraitsT > **_NFAT**
- typedef basic_regex< _CharT, _TraitsT > **_RegexT**
- typedef ::vector< sub_match< _BiIter >, _Alloc > **_ResultsVec**

Public Member Functions

- **_Executor** (_BiIter __begin, _BiIter __end, _ResultsVec &__results, const _RegexT &__re, _FlagT __flags)
- bool **_M_match** ()
- bool **_M_search** ()
- bool **_M_search_from_first** ()

Public Attributes

- _BiIter **_M_begin**
- _ResultsVec **_M_cur_results**
- _BiIter **_M_current**
- const _BiIter **_M_end**
- _FlagT **_M_flags**
- bool **_M_has_sol**

- const _NFAT & **_M_nfa**
- const _RegexT & **_M_re**
- ::vector< pair< _Bilter, int > > **_M_rep_count**
- _ResultsVec & **_M_results**
- _State_info< __search_mode, _ResultsVec > **_M_states**

5.81.1 Detailed Description

template<typename _Bilter, typename _Alloc, typename _TraitsT, bool __dfs_mode>
class std::__detail::_Executor< _Bilter, _Alloc, _TraitsT, __dfs_mode >

Takes a regex and an input string and does the matching.
The _Executor class has two modes: DFS mode and BFS mode, controlled by the template parameter __dfs_mode.
The documentation for this class was generated from the following files:

- regex.h
- regex_executor.h
- regex_executor.tcc

5.82 __gnu_cxx::_ExtPtr_allocator< _Tp > Class Template Reference

`#include <extptr_allocator.h>`

Public Types

- typedef _Pointer_adapter< _Relative_pointer_impl< const _Tp > > **const_pointer**
- typedef const _Tp & **const_reference**
- typedef std::ptrdiff_t **difference_type**
- typedef _Pointer_adapter< _Relative_pointer_impl< _Tp > > **pointer**
- typedef _Tp & **reference**
- typedef std::size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **_ExtPtr_allocator** (const _ExtPtr_allocator &__rarg) noexcept
- template<typename _Up>
  _ExtPtr_allocator (const _ExtPtr_allocator< _Up > &__rarg) noexcept
- const std::allocator< _Tp > & **_M_getUnderlyingImp** () const
- const_pointer **address** (const_reference __x) const noexcept
- pointer **address** (reference __x) const noexcept
- pointer **allocate** (size_type __n, const void *=0)
- template<typename _Up, typename... _Args>
  void **construct** (_Up *__p, _Args &&... __args)
- template<typename... _Args>
  void **construct** (pointer __p, _Args &&... __args)
- void **deallocate** (pointer __p, size_type __n)
- template<typename _Up>
  void **destroy** (_Up *__p)
- void **destroy** (pointer __p)
- size_type **max_size** () const noexcept
- bool **operator==** (const _ExtPtr_allocator &__rarg) const
- template<typename _Up>
  bool **operator==** (const _ExtPtr_allocator< _Up > &__rarg) const

Friends

- template<typename _Up>
  void **swap** (_ExtPtr_allocator< _Up > &, _ExtPtr_allocator< _Up > &)

5.82.1 Detailed Description

template<typename _Tp>
class __gnu_cxx::_ExtPtr_allocator< _Tp >

An example allocator which uses a non-standard pointer type.
This allocator specifies that containers use a 'relative pointer' as it's pointer type. (See ext/pointer.h) Memory allocation in this example is still performed using std::allocator.
The documentation for this class was generated from the following file:

- extptr_allocator.h

5.83 __gnu_cxx::__detail::_Ffit_finder< _Tp > Class Template Reference

`#include <bitmap_allocator.h>`

Public Types

- typedef _Block_pair **argument_type**
- typedef bool **result_type**

Public Member Functions

- std::size_t * **_M_get** () const throw ()
- _Counter_type **_M_offset** () const throw ()
- bool **operator()** (_Block_pair __bp) throw ()

5.83.1 Detailed Description

template<typename _Tp>
class __gnu_cxx::__detail::_Ffit_finder< _Tp >

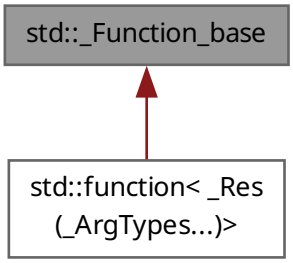
The class which acts as a predicate for applying the first-fit memory allocation policy for the bitmap allocator.
The documentation for this class was generated from the following file:

- bitmap_allocator.h

5.84 std::_Function_base Class Reference

`#include <std_function.h>`

Inheritance diagram for std::_Function_base:



Public Types

- using **_Manager_type**

Public Member Functions

- bool **_M_empty** () const

Public Attributes

- _Any_data **_M_functor**
- _Manager_type **_M_manager**

Static Public Attributes

- static const size_t **_M_max_align**
- static const size_t **_M_max_size**

5.84.1 Detailed Description

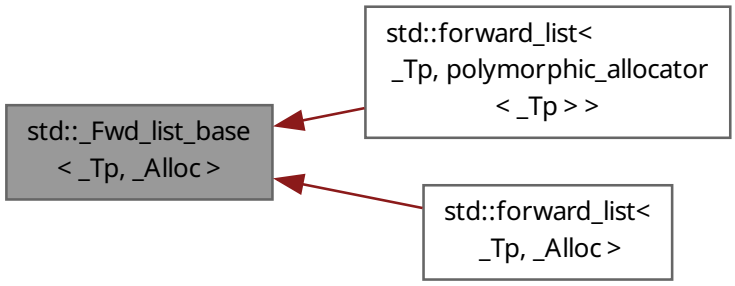
Base class of all polymorphic function object wrappers.
The documentation for this class was generated from the following file:

- std_function.h

5.85 std::_Fwd_list_base< _Tp, _Alloc > Struct Template Reference

`#include <forward_list.h>`

Inheritance diagram for std::_Fwd_list_base< _Tp, _Alloc >:



Public Types

- using **const_iterator**
- using **iterator**
- using **pointer**

Public Member Functions

- **_Fwd_list_base** (_Fwd_list_base &&)=default
- **_Fwd_list_base** (_Fwd_list_base &&__lst, _Node_alloc_type &&__a)
- **_Fwd_list_base** (_Fwd_list_base &&__lst, _Node_alloc_type &&__a, std::true_type)
- **_Fwd_list_base** (_Node_alloc_type &&__a)
- const _Node_alloc_type & **_M_get_Node_allocator** () const noexcept
- _Node_alloc_type & **_M_get_Node_allocator** () noexcept
- template<typename... _Args>
  auto **_M_insert_after** (const_iterator __pos, _Args &&... __args) -> _Base_ptr

Protected Types

- using **_Base_ptr**
- using **_Node**
- using **_Node_alloc_traits**
- using **_Node_alloc_type**
- using **_Node_ptr**
- using **_Node_traits**

Protected Member Functions

- template<typename... _Args>
  _Node_ptr **_M_create_node** (_Args &&... __args)
- void **_M_destroy_node** (_Node_ptr __p)
- _Base_ptr **_M_erase_after** (_Base_ptr __pos)
- _Base_ptr **_M_erase_after** (_Base_ptr __pos, _Base_ptr __last)
- _Node * **_M_get_node** ()

- template<typename... _Args>
  _Base_ptr **_M_insert_after** (const_iterator __pos, _Args &&... __args)
- void **_M_put_node** (_Node_ptr __p)

Protected Attributes

- _Fwd_list_impl **_M_impl**

5.85.1 Detailed Description

template<typename _Tp, typename _Alloc>
struct std::_Fwd_list_base< _Tp, _Alloc >

Base class for forward_list.
The documentation for this struct was generated from the following files:

- forward_list.h
- forward_list.tcc

5.86 std::_Fwd_list_const_iterator< _Tp > Struct Template Reference

`#include <forward_list.h>`

Public Types

- typedef const _Fwd_list_node< _Tp > **_Node**
- typedef _Fwd_list_const_iterator< _Tp > **_Self**
- typedef ptrdiff_t **difference_type**
- typedef _Fwd_list_iterator< _Tp > **iterator**
- typedef std::forward_iterator_tag **iterator_category**
- typedef const _Tp * **pointer**
- typedef const _Tp & **reference**
- typedef _Tp **value_type**

Public Member Functions

- **_Fwd_list_const_iterator** (const _Fwd_list_node_base *__n) noexcept
- **_Fwd_list_const_iterator** (const iterator &__iter) noexcept
- reference **operator*** () const noexcept
- _Self & **operator++** () noexcept
- _Self **operator++** (int) noexcept
- pointer **operator->** () const noexcept

Friends

- template<typename, typename>
  struct **_Fwd_list_base**
- template<typename, typename>
  class **forward_list**
- bool operator== (const _Self &__x, const _Self &__y) noexcept

5.86.1 Detailed Description

template<typename _Tp>
struct std::_Fwd_list_const_iterator< _Tp >

A forward_list::const_iterator.
All the functions are op overloads.

5.86.2 Friends And Related Symbol Documentation

operator==

```
template<typename _Tp>
bool operator== (
            const _Self & __x,
            const _Self & __y)  [friend]
```

Forward list const_iterator equality comparison.
The documentation for this struct was generated from the following file:

- forward_list.h

5.87 std::_Fwd_list_iterator< _Tp > Struct Template Reference

`#include <forward_list.h>`

Public Types

- typedef _Fwd_list_node< _Tp > **_Node**
- typedef _Fwd_list_iterator< _Tp > **_Self**
- typedef ptrdiff_t **difference_type**
- typedef std::forward_iterator_tag **iterator_category**
- typedef _Tp * **pointer**
- typedef _Tp & **reference**
- typedef _Tp **value_type**

Public Member Functions

- **_Fwd_list_iterator** (_Fwd_list_node_base *__n) noexcept
- reference **operator*** () const noexcept
- _Self & **operator++** () noexcept
- _Self **operator++** (int) noexcept
- pointer **operator->** () const noexcept

Friends

- template<typename, typename>
  struct **_Fwd_list_base**
- struct **_Fwd_list_const_iterator< _Tp >**
- template<typename, typename>
  class **forward_list**
- bool operator== (const _Self &__x, const _Self &__y) noexcept

5.87.1 Detailed Description

template<typename _Tp>
struct std::_Fwd_list_iterator< _Tp >

A forward_list::iterator.
All the functions are op overloads.

5.87.2 Friends And Related Symbol Documentation

operator==

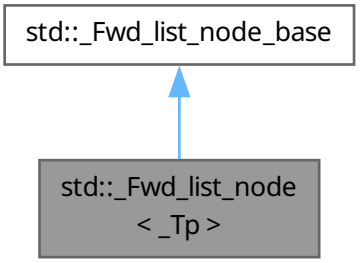
```
template<typename _Tp>
bool operator== (
            const _Self & __x,
            const _Self & __y)  [friend]
```

Forward list iterator equality comparison.
The documentation for this struct was generated from the following file:

- forward_list.h

5.88 std::_Fwd_list_node< _Tp > Struct Template Reference

`#include <forward_list.h>`
Inheritance diagram for std::_Fwd_list_node< _Tp >:



Public Types

- using **_Base_ptr**
- using **_Node_ptr**

Public Member Functions

- _Fwd_list_node_base * **_M_base_ptr** ()
- const _Fwd_list_node_base * **_M_base_ptr** () const
- _Node_ptr **_M_node_ptr** ()
- void **_M_reverse_after** () noexcept
- _Fwd_list_node_base * **_M_transfer_after** (_Fwd_list_node_base *__begin, _Fwd_list_node_base *__end) noexcept

- const _Tp * **_M_valptr** () const noexcept
- _Tp * **_M_valptr** () noexcept

Public Attributes

- _Fwd_list_node_base * **_M_next**
- __gnu_cxx::__aligned_buffer< _Tp > **_M_storage**

5.88.1 Detailed Description

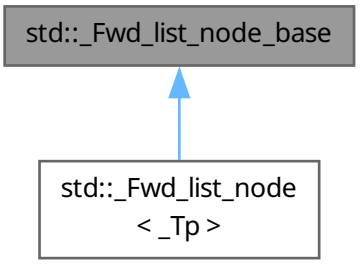
template<typename _Tp>
struct std::_Fwd_list_node< _Tp >

A helper node class for forward_list. This is just a linked list with uninitialized storage for a data value in each node. There is a sorting utility method.
The documentation for this struct was generated from the following file:

- forward_list.h

5.89 std::_Fwd_list_node_base Struct Reference

`#include <forward_list.h>`
Inheritance diagram for std::_Fwd_list_node_base:



Public Types

- using **_Base_ptr**

Public Member Functions

- **_Fwd_list_node_base** (_Fwd_list_node_base &&__x) noexcept
- **_Fwd_list_node_base** (const _Fwd_list_node_base &)=delete
- _Fwd_list_node_base * **_M_base_ptr** ()
- const _Fwd_list_node_base * **_M_base_ptr** () const
- void **_M_reverse_after** () noexcept
- _Fwd_list_node_base * **_M_transfer_after** (_Fwd_list_node_base *__begin, _Fwd_list_node_base *__end) noexcept
- _Fwd_list_node_base & **operator=** (_Fwd_list_node_base &&__x) noexcept
- _Fwd_list_node_base & **operator=** (const _Fwd_list_node_base &)=delete

Public Attributes

- _Fwd_list_node_base * **_M_next**

5.89.1 Detailed Description

A helper basic node class for forward_list.

```
This is just a linked list with nothing inside it.
There are purely list shuffling utility methods here.
```

The documentation for this struct was generated from the following file:

- forward_list.h

5.90 __gnu_parallel::_GuardedIterator< _RAIter, _Compare > Class Template Reference

`#include <multiway_merge.h>`

Public Member Functions

- _GuardedIterator (_RAIter __begin, _RAIter __end, _Compare &__comp)
- operator _RAIter () const
- std::iterator_traits< _RAIter >::value_type & operator* () const
- _GuardedIterator< _RAIter, _Compare > & operator++ ()

Friends

- bool operator< (const _GuardedIterator< _RAIter, _Compare > &__bi1, const _GuardedIterator< _RAIter, _Compare > &__bi2)
- bool operator<= (const _GuardedIterator< _RAIter, _Compare > &__bi1, const _GuardedIterator< _RAIter, _Compare > &__bi2)

5.90.1 Detailed Description

template<typename _RAIter, typename _Compare>
class __gnu_parallel::_GuardedIterator< _RAIter, _Compare >

_Iterator wrapper supporting an implicit supremum at the end of the sequence, dominating all comparisons.
The implicit supremum comes with a performance cost.
Deriving from _RAIter is not possible since _RAIter need not be a class.

5.90.2 Constructor & Destructor Documentation

_GuardedIterator()

```
template<typename _RAIter, typename _Compare>
__gnu_parallel::_GuardedIterator< _RAIter, _Compare >::_GuardedIterator (
            _RAIter __begin,
            _RAIter __end,
            _Compare & __comp)  [inline]
```

Constructor. Sets iterator to beginning of sequence.

Parameters

__begin	Begin iterator of sequence.
__end	End iterator of sequence.

__comp	Comparator provided for associated overloaded compare operators.

Referenced by operator<, and operator<=.

5.90.3 Member Function Documentation

operator _RAIter()

```
template<typename _RAIter, typename _Compare>
__gnu_parallel::_GuardedIterator< _RAIter, _Compare >::operator _RAIter () const  [inline]
```

Convert to wrapped iterator.

Returns

Wrapped iterator.

operator*()

```
template<typename _RAIter, typename _Compare>
std::iterator_traits< _RAIter >::value_type & __gnu_parallel::_GuardedIterator< _RAIter, _Compare
>::operator* () const  [inline]
```

Dereference operator.

Returns

Referenced element.

operator++()

```
template<typename _RAIter, typename _Compare>
_GuardedIterator< _RAIter, _Compare > & __gnu_parallel::_GuardedIterator< _RAIter, _Compare >↩
::operator++ ()  [inline]
```

Pre-increment operator.

Returns

This.

5.90.4 Friends And Related Symbol Documentation

operator<

```
template<typename _RAIter, typename _Compare>
bool operator< (
            const _GuardedIterator< _RAIter, _Compare > & __bi1,
            const _GuardedIterator< _RAIter, _Compare > & __bi2)  [friend]
```

Compare two elements referenced by guarded iterators.

Parameters

__bi1	First iterator.
__bi2	Second iterator.

Returns

`true` if less.

References _GuardedIterator().

operator<=

```
template<typename _RAIter, typename _Compare>
bool operator<= (
            const _GuardedIterator< _RAIter, _Compare > & __bi1,
            const _GuardedIterator< _RAIter, _Compare > & __bi2)  [friend]
```

Compare two elements referenced by guarded iterators.

Parameters

__bi1	First iterator.
__bi2	Second iterator.

Returns

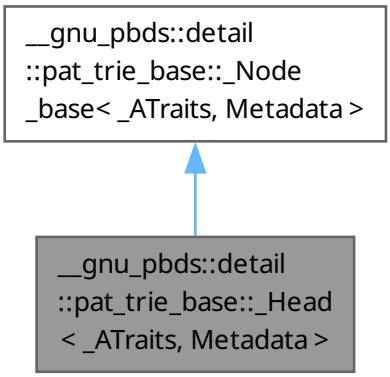
`True` if less equal.

References _GuardedIterator().
The documentation for this class was generated from the following file:

- multiway_merge.h

5.91 __gnu_pbds::detail::pat_trie_base::_Head< _ATraits, Metadata > Struct Template Reference

`#include <pat_trie_base.hpp>`
Inheritance diagram for __gnu_pbds::detail::pat_trie_base::_Head< _ATraits, Metadata >:



Public Types

- typedef _ATraits::const_iterator **a_const_iterator**
- typedef detail::rebind_traits< _Alloc, _ATraits >::const_pointer **a_const_pointer**
- typedef _ATraits **access_traits**
- typedef _Alloc **allocator_type**

- typedef _Node_base< _ATraits, Metadata > **base_type**
- typedef base_type::node_pointer **node_pointer**
- typedef base_type::type_traits **type_traits**

Public Attributes

- node_pointer **m_p_max**
- node_pointer **m_p_min**
- node_pointer **m_p_parent**
- const node_type **m_type**

5.91.1 Detailed Description

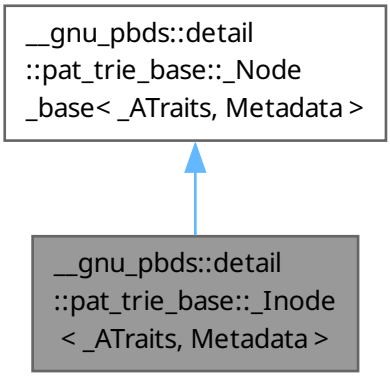
template<typename _ATraits, typename Metadata>
struct __gnu_pbds::detail::pat_trie_base::_Head< _ATraits, Metadata >

Head node for PATRICIA tree.
The documentation for this struct was generated from the following file:

- pat_trie_base.hpp

5.92 __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata > Struct Template Reference

`#include <pat_trie_base.hpp>`
Inheritance diagram for __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >:



Classes

- struct const_iterator
- struct iterator

Public Types

- enum { **arr_size** }
- typedef detail::rebind_traits< _Alloc, node_pointer > **__rebind_np**
- typedef base_type::allocator_type **_Alloc**
- typedef base_type::access_traits **access_traits**
- typedef _Alloc **allocator_type**
- typedef _Node_base< _ATraits, Metadata > **base_type**
- typedef __rebind_np::pointer **node_pointer_pointer**
- typedef __rebind_np::reference **node_pointer_reference**
- typedef _Alloc::size_type **size_type**
- typedef base_type::type_traits **type_traits**
- typedef type_traits::value_type **value_type**

Public Member Functions

- **_Inode** (size_type, const a_const_iterator)
- node_pointer **add_child** (node_pointer, a_const_iterator, a_const_iterator, a_const_pointer)
- iterator **begin** ()
- const_iterator **begin** () const
- iterator **end** ()
- const_iterator **end** () const
- iterator **get_child_it** (a_const_iterator, a_const_iterator, a_const_pointer)
- node_pointer **get_child_node** (a_const_iterator, a_const_iterator, a_const_pointer)
- node_const_pointer **get_child_node** (a_const_iterator, a_const_iterator, a_const_pointer) const
- size_type **get_e_ind** () const
- node_const_pointer **get_join_child** (node_const_pointer, a_const_pointer) const
- node_pointer **get_join_child** (node_pointer, a_const_pointer)
- node_pointer **get_lower_bound_child_node** (a_const_iterator, a_const_iterator, size_type, a_const_pointer)
- leaf_pointer **leftmost_descendant** ()
- leaf_const_pointer **leftmost_descendant** () const
- a_const_iterator **pref_b_it** () const
- a_const_iterator **pref_e_it** () const
- void **remove_child** (iterator)
- void **remove_child** (node_pointer)
- void **replace_child** (node_pointer, a_const_iterator, a_const_iterator, a_const_pointer)
- leaf_pointer **rightmost_descendant** ()
- leaf_const_pointer **rightmost_descendant** () const
- bool **should_be_mine** (a_const_iterator, a_const_iterator, size_type, a_const_pointer) const
- void **update_prefixes** (a_const_pointer)

Public Attributes

- node_pointer **m_p_parent**
- const node_type **m_type**

5.92.1 Detailed Description

template<typename _ATraits, typename Metadata>
struct __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >

Internal node type, PATRICIA tree.
The documentation for this struct was generated from the following file:

- pat_trie_base.hpp

5.93 __gnu_cxx::_Invalid_type Struct Reference

`#include <pointer.h>`

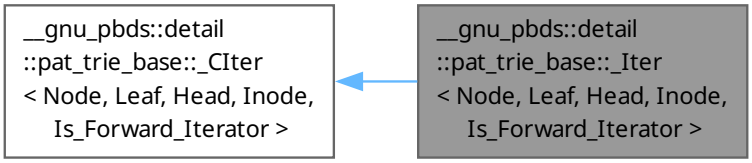
5.93.1 Detailed Description

The specialization on this type helps resolve the problem of reference to void, and eliminates the need to specialize _Pointer_adapter for cases of void*, const void*, and so on.
The documentation for this struct was generated from the following file:

- pointer.h

5.94 __gnu_pbds::detail::pat_trie_base::_Iter< Node, Leaf, Head, Inode, Is_Forward_Iterator > Class Template Reference

`#include <pat_trie_base.hpp>`
Inheritance diagram for __gnu_pbds::detail::pat_trie_base::_Iter< Node, Leaf, Head, Inode, Is_Forward_Iterator >:



Public Types

- typedef allocator_type **_Alloc**
- typedef base_type::allocator_type **allocator_type**
- typedef _CIter< Node, Leaf, Head, Inode, Is_Forward_Iterator > **base_type**
- typedef type_traits::const_pointer **const_pointer**
- typedef type_traits::const_reference **const_reference**
- typedef allocator_type::difference_type **difference_type**
- typedef base_type::head_pointer **head_pointer**
- typedef Inode::iterator **inode_iterator**
- typedef base_type::inode_pointer **inode_pointer**
- typedef std::bidirectional_iterator_tag **iterator_category**
- typedef base_type::leaf_const_pointer **leaf_const_pointer**
- typedef base_type::leaf_pointer **leaf_pointer**
- typedef base_type::node_pointer **node_pointer**
- typedef type_traits::pointer **pointer**
- typedef type_traits::reference **reference**
- typedef base_type::type_traits **type_traits**
- typedef type_traits::value_type **value_type**

Public Member Functions

- **_Iter** (const _Iter< Node, Leaf, Head, Inode, !Is_Forward_Iterator > &other)
- **_Iter** (node_pointer p_nd=0)
- bool **operator!=** (const _CIter &other) const
- bool **operator!=** (const _CIter< Node, Leaf, Head, Inode, !Is_Forward_Iterator > &other) const
- reference **operator*** () const
- _Iter & **operator++** ()
- _Iter **operator++** (int)
- _Iter & **operator--** ()
- _Iter **operator--** (int)
- pointer **operator->** () const
- _Iter & **operator=** (const _Iter &other)
- _Iter & **operator=** (const _Iter< Node, Leaf, Head, Inode, !Is_Forward_Iterator > &other)
- bool **operator==** (const _CIter &other) const
- bool **operator==** (const _CIter< Node, Leaf, Head, Inode, !Is_Forward_Iterator > &other) const

Public Attributes

- node_pointer **m_p_nd**

Protected Member Functions

- void **dec** (false_type)
- void **dec** (true_type)
- void **inc** (false_type)
- void **inc** (true_type)

Static Protected Member Functions

- static node_pointer **get_larger_sibling** (node_pointer p_nd)
- static node_pointer **get_smaller_sibling** (node_pointer p_nd)
- static leaf_pointer **leftmost_descendant** (node_pointer p_nd)
- static leaf_pointer **rightmost_descendant** (node_pointer p_nd)

5.94.1 Detailed Description

template<typename Node, typename Leaf, typename Head, typename Inode, bool Is_Forward_Iterator>
class __gnu_pbds::detail::pat_trie_base::_Iter< Node, Leaf, Head, Inode, Is_Forward_Iterator >

Iterator.
The documentation for this class was generated from the following file:

- pat_trie_base.hpp

5.95 std::__fwdlist::_Iterator< _Const, _Ptr > Class Template Reference

`#include <forward_list.h>`

Public Types

- using **difference_type**
- using **iterator_category**
- using **pointer**
- using **reference**
- using **value_type**

Public Member Functions

- constexpr **_Iterator** (_Base_ptr __x) noexcept
- **_Iterator** (const _Iterator &)=default
- constexpr **_Iterator** (const _Iterator< false, _Ptr > &__i)
- constexpr reference **operator*** () const noexcept
- constexpr _Iterator & **operator++** () noexcept
- constexpr _Iterator **operator++** (int) noexcept
- constexpr pointer **operator->** () const noexcept
- _Iterator & **operator=** (const _Iterator &)=default

Friends

- template<typename _Tp, typename _Allocator>
  struct **::_Fwd_list_base**
- template<typename _Tp, typename _Allocator>
  class **::forward_list**
- constexpr bool operator== (const _Iterator &__x, const _Iterator &__y) noexcept

5.95.1 Detailed Description

template<bool _Const, typename _Ptr>
class std::__fwdlist::_Iterator< _Const, _Ptr >

A forward_list iterator when the allocator uses fancy pointers.

5.95.2 Friends And Related Symbol Documentation

operator==

```
template<bool _Const, typename _Ptr>
bool operator== (
            const _Iterator< _Const, _Ptr > & __x,
            const _Iterator< _Const, _Ptr > & __y)  [friend]
```

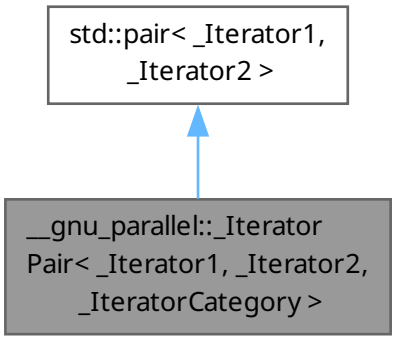
Forward list iterator equality comparison.
The documentation for this class was generated from the following file:

- forward_list.h

5.96 __gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory > Class Template Reference

```
#include <iterator.h>
```

Inheritance diagram for __gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >:



Public Types

- typedef std::iterator_traits< _Iterator1 > **_TraitsType**
- typedef _TraitsType::difference_type **difference_type**
- typedef _Iterator1 first_type
- typedef _Iterator1 first_type
- typedef _Iterator1 first_type
- typedef _IteratorCategory **iterator_category**
- typedef _IteratorPair * **pointer**
- typedef _IteratorPair & **reference**
- typedef _Iterator2 second_type
- typedef _Iterator2 second_type
- typedef _Iterator2 second_type
- typedef void **value_type**

Public Member Functions

- **_IteratorPair** (const _Iterator1 &__first, const _Iterator2 &__second)
- **operator _Iterator2** () const
- _IteratorPair **operator+** (difference_type __delta) const
- _IteratorPair & **operator++** ()
- const _IteratorPair **operator++** (int)
- difference_type **operator-** (const _IteratorPair &__other) const
- _IteratorPair & **operator--** ()
- const _IteratorPair **operator--** (int)
- _IteratorPair & **operator=** (const _IteratorPair &__other)
- constexpr void swap (pair &__p) noexcept(__and_< __is_nothrow_swappable< _Iterator1 >, __is_nothrow_↩ swappable< _Iterator2 > >::value)
- constexpr void swap (pair &__p) noexcept(__and_< __is_nothrow_swappable< _Iterator1 >, __is_nothrow_↩ swappable< _Iterator2 > >::value)
- constexpr void swap (pair &__p) noexcept(__and_< __is_nothrow_swappable< _Iterator1 >, __is_nothrow_↩ swappable< _Iterator2 > >::value)
- constexpr void swap (pair &__p) noexcept(__and_< __is_nothrow_swappable< _Iterator1 >, __is_nothrow_↩ swappable< _Iterator2 > >::value)

Public Attributes

- _Iterator1 first
- _Iterator1 first
- _Iterator1 first
- _Iterator1 first
- _Iterator2 second
- _Iterator2 second
- _Iterator2 second
- _Iterator2 second

Related Symbols

(Note that these are not member symbols.)

- constexpr enable_if< __and_< __is_swappable< _Iterator1 >, __is_swappable< _Iterator2 > >::value >::type swap (pair< _Iterator1, _Iterator2 > &__x, pair< _Iterator1, _Iterator2 > &__y) noexcept(noexcept(__x.swap(↩ __y)))
- constexpr enable_if< __and_< __is_swappable< _Iterator1 >, __is_swappable< _Iterator2 > >::value >::type swap (pair< _Iterator1, _Iterator2 > &__x, pair< _Iterator1, _Iterator2 > &__y) noexcept(noexcept(__x.swap(↩ __y)))
- constexpr enable_if< __and_< __is_swappable< _Iterator1 >, __is_swappable< _Iterator2 > >::value >::type swap (pair< _Iterator1, _Iterator2 > &__x, pair< _Iterator1, _Iterator2 > &__y) noexcept(noexcept(__x.swap(↩ __y)))
- constexpr enable_if< __and_< __is_swappable< _Iterator1 >, __is_swappable< _Iterator2 > >::value >::type swap (pair< _Iterator1, _Iterator2 > &__x, pair< _Iterator1, _Iterator2 > &__y) noexcept(noexcept(__x.swap(↩ __y)))
- constexpr bool operator== (const pair< _Iterator1, _Iterator2 > &__x, const pair< _U1, _U2 > &__y)
- constexpr bool operator== (const pair< _Iterator1, _Iterator2 > &__x, const pair< _U1, _U2 > &__y)
- constexpr bool operator== (const pair< _Iterator1, _Iterator2 > &__x, const pair< _U1, _U2 > &__y)
- constexpr bool operator== (const pair< _Iterator1, _Iterator2 > &__x, const pair< _U1, _U2 > &__y)
- constexpr common_comparison_category_t< __detail::__synth3way_t< _Iterator1, _U1 >, __detail::__↩ synth3way_t< _Iterator2, _U2 > > operator<=> (const pair< _Iterator1, _Iterator2 > &__x, const pair< _U1, _U2 > &__y)
- constexpr common_comparison_category_t< __detail::__synth3way_t< _Iterator1, _U1 >, __detail::__↩ synth3way_t< _Iterator2, _U2 > > operator<=> (const pair< _Iterator1, _Iterator2 > &__x, const pair< _U1, _U2 > &__y)
- constexpr common_comparison_category_t< __detail::__synth3way_t< _Iterator1, _U1 >, __detail::__↩ synth3way_t< _Iterator2, _U2 > > operator<=> (const pair< _Iterator1, _Iterator2 > &__x, const pair< _U1, _U2 > &__y)
- constexpr common_comparison_category_t< __detail::__synth3way_t< _Iterator1, _U1 >, __detail::__↩ synth3way_t< _Iterator2, _U2 > > operator<=> (const pair< _Iterator1, _Iterator2 > &__x, const pair< _U1, _U2 > &__y)

5.96.1 Detailed Description

template<typename _Iterator1, typename _Iterator2, typename _IteratorCategory>
class __gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >

A pair of iterators. The usual iterator operations are applied to both child iterators.

5.96.2 Member Typedef Documentation

first_type `[1/3]`

```
typedef _Iterator1 std::pair< _Iterator1, _Iterator2 >::first_type  [inherited]
```

The type of the `first` member.

first_type [2/3]

typedef _Iterator1 std::pair< _Iterator1, _Iterator2 >::first_type [inherited]
The type of the first member.

first_type [3/3]

typedef _Iterator1 std::pair< _Iterator1, _Iterator2 >::first_type [inherited]
The type of the first member.

second_type [1/3]

typedef _Iterator2 std::pair< _Iterator1, _Iterator2 >::second_type [inherited]
The type of the second member.

second_type [2/3]

typedef _Iterator2 std::pair< _Iterator1, _Iterator2 >::second_type [inherited]
The type of the second member.

second_type [3/3]

typedef _Iterator2 std::pair< _Iterator1, _Iterator2 >::second_type [inherited]
The type of the second member.

5.96.3 Member Function Documentation

swap() [1/4]

void std::pair< _Iterator1, _Iterator2 >::swap (
pair< _Iterator1, _Iterator2 > & __p) [inline], [constexpr], [noexcept], [inherited]
Swap the first members and then the second members.

swap() [2/4]

void std::pair< _Iterator1, _Iterator2 >::swap (
pair< _Iterator1, _Iterator2 > & __p) [inline], [constexpr], [noexcept], [inherited]
Swap the first members and then the second members.

swap() [3/4]

void std::pair< _Iterator1, _Iterator2 >::swap (
pair< _Iterator1, _Iterator2 > & __p) [inline], [constexpr], [noexcept], [inherited]
Swap the first members and then the second members.

swap() [4/4]

void std::pair< _Iterator1, _Iterator2 >::swap (
pair< _Iterator1, _Iterator2 > & __p) [inline], [constexpr], [noexcept], [inherited]
Swap the first members and then the second members.

5.96.4 Friends And Related Symbol Documentation

operator<=>() [1/4]

common_comparison_category_t< __detail::__synth3way_t< _Iterator1, _U1 >, __detail::__synth3way↩
_t< _Iterator2, _U2 > > operator<=> (

```
            const pair< _Iterator1, _Iterator2 > & __x,
            const pair< _U1, _U2 > & __y)  [related]
```

Defines a lexicographical order for pairs.
For two pairs of comparable types, `P` is ordered before `Q` if `P.first` is less than `Q.first`, or if `P.first` and `Q.first` are equivalent (neither is less than the other) and `P.second` is less than `Q.second`.

operator<=>() [2/4]

```
common_comparison_category_t< __detail::__synth3way_t< _Iterator1, _U1 >, __detail::__synth3way↩
_t< _Iterator2, _U2 > > operator<=> (
            const pair< _Iterator1, _Iterator2 > & __x,
            const pair< _U1, _U2 > & __y)  [related]
```

Defines a lexicographical order for pairs.
For two pairs of comparable types, `P` is ordered before `Q` if `P.first` is less than `Q.first`, or if `P.first` and `Q.first` are equivalent (neither is less than the other) and `P.second` is less than `Q.second`.

operator<=>() [3/4]

```
common_comparison_category_t< __detail::__synth3way_t< _Iterator1, _U1 >, __detail::__synth3way↩
_t< _Iterator2, _U2 > > operator<=> (
            const pair< _Iterator1, _Iterator2 > & __x,
            const pair< _U1, _U2 > & __y)  [related]
```

Defines a lexicographical order for pairs.
For two pairs of comparable types, `P` is ordered before `Q` if `P.first` is less than `Q.first`, or if `P.first` and `Q.first` are equivalent (neither is less than the other) and `P.second` is less than `Q.second`.

operator<=>() [4/4]

```
common_comparison_category_t< __detail::__synth3way_t< _Iterator1, _U1 >, __detail::__synth3way↩
_t< _Iterator2, _U2 > > operator<=> (
            const pair< _Iterator1, _Iterator2 > & __x,
            const pair< _U1, _U2 > & __y)  [related]
```

Defines a lexicographical order for pairs.
For two pairs of comparable types, `P` is ordered before `Q` if `P.first` is less than `Q.first`, or if `P.first` and `Q.first` are equivalent (neither is less than the other) and `P.second` is less than `Q.second`.

operator==() [1/4]

```
bool operator== (
            const pair< _Iterator1, _Iterator2 > & __x,
            const pair< _U1, _U2 > & __y)  [related]
```

Two pairs are equal iff their members are equal.

operator==() [2/4]

```
bool operator== (
            const pair< _Iterator1, _Iterator2 > & __x,
            const pair< _U1, _U2 > & __y)  [related]
```

Two pairs are equal iff their members are equal.

operator==() [3/4]

```
bool operator== (
            const pair< _Iterator1, _Iterator2 > & __x,
            const pair< _U1, _U2 > & __y)  [related]
```

Two pairs are equal iff their members are equal.

operator==() [4/4]

```
bool operator== (
            const pair< _Iterator1, _Iterator2 > & __x,
            const pair< _U1, _U2 > & __y)  [related]
```

Two pairs are equal iff their members are equal.

swap() [1/4]

```
enable_if< __and_< __is_swappable< _Iterator1 >, __is_swappable< _Iterator2 > >::value >::type
swap (
            pair< _Iterator1, _Iterator2 > & __x,
            pair< _Iterator1, _Iterator2 > & __y)  [related]
```

Swap overload for pairs. Calls std::pair::swap().

Note

This std::swap overload is not declared in C++03 mode, which has performance implications, e.g. see https://gcc.gnu.org/PR38466

swap() [2/4]

```
enable_if< __and_< __is_swappable< _Iterator1 >, __is_swappable< _Iterator2 > >::value >::type
swap (
            pair< _Iterator1, _Iterator2 > & __x,
            pair< _Iterator1, _Iterator2 > & __y)  [related]
```

Swap overload for pairs. Calls std::pair::swap().

Note

This std::swap overload is not declared in C++03 mode, which has performance implications, e.g. see https://gcc.gnu.org/PR38466

swap() [3/4]

```
enable_if< __and_< __is_swappable< _Iterator1 >, __is_swappable< _Iterator2 > >::value >::type
swap (
            pair< _Iterator1, _Iterator2 > & __x,
            pair< _Iterator1, _Iterator2 > & __y)  [related]
```

Swap overload for pairs. Calls std::pair::swap().

Note

This std::swap overload is not declared in C++03 mode, which has performance implications, e.g. see https://gcc.gnu.org/PR38466

swap() [4/4]

```
enable_if< __and_< __is_swappable< _Iterator1 >, __is_swappable< _Iterator2 > >::value >::type
swap (
            pair< _Iterator1, _Iterator2 > & __x,
            pair< _Iterator1, _Iterator2 > & __y)  [related]
```

Swap overload for pairs. Calls std::pair::swap().

Note

This std::swap overload is not declared in C++03 mode, which has performance implications, e.g. see https://gcc.gnu.org/PR38466

5.96.5 Member Data Documentation

first [1/4]

```
_Iterator1 std::pair< _Iterator1, _Iterator2 >::first  [inherited]
```

The first member.

first [2/4]

```
_Iterator1 std::pair< _Iterator1, _Iterator2 >::first  [inherited]
```

The first member.

first [3/4]

```
_Iterator1 std::pair< _Iterator1, _Iterator2 >::first  [inherited]
```

The first member.

first [4/4]

```
_Iterator1 std::pair< _Iterator1, _Iterator2 >::first  [inherited]
```

The first member.

second [1/4]

```
_Iterator2 std::pair< _Iterator1, _Iterator2 >::second  [inherited]
```

The second member.

second [2/4]

```
_Iterator2 std::pair< _Iterator1, _Iterator2 >::second  [inherited]
```

The second member.

second [3/4]

```
_Iterator2 std::pair< _Iterator1, _Iterator2 >::second  [inherited]
```

The second member.

second [4/4]

```
_Iterator2 std::pair< _Iterator1, _Iterator2 >::second  [inherited]
```

The second member.

The documentation for this class was generated from the following file:

- iterator.h

5.97 __gnu_parallel::_IteratorTriple< _Iterator1, _Iterator2, _Iterator3, _IteratorCategory > Class Template Reference

```
#include <iterator.h>
```

Public Types

- typedef std::iterator_traits< _Iterator1 >::difference_type **difference_type**
- typedef _IteratorCategory **iterator_category**
- typedef _IteratorTriple * **pointer**
- typedef _IteratorTriple & **reference**
- typedef void **value_type**

Public Member Functions

- **_IteratorTriple** (const _Iterator1 &__first, const _Iterator2 &__second, const _Iterator3 &__third)
- **operator _Iterator3** () const
- _IteratorTriple **operator+** (difference_type __delta) const
- _IteratorTriple & **operator++** ()
- const _IteratorTriple **operator++** (int)
- difference_type **operator-** (const _IteratorTriple &__other) const
- _IteratorTriple & **operator--** ()
- const _IteratorTriple **operator--** (int)
- _IteratorTriple & **operator=** (const _IteratorTriple &__other)

Public Attributes

- _Iterator1 **_M_first**
- _Iterator2 **_M_second**
- _Iterator3 **_M_third**

5.97.1 Detailed Description

template<typename _Iterator1, typename _Iterator2, typename _Iterator3, typename _IteratorCategory>
class __gnu_parallel::_IteratorTriple< _Iterator1, _Iterator2, _Iterator3, _IteratorCategory >

A triple of iterators. The usual iterator operations are applied to all three child iterators.
The documentation for this class was generated from the following file:

- iterator.h

5.98 __gnu_parallel::_Job< _DifferenceTp > Struct Template Reference

`#include <workstealing.h>`

Public Types

- typedef _DifferenceTp **_DifferenceType**

Public Attributes

- volatile _DifferenceType _M_first
- volatile _DifferenceType _M_last
- volatile _DifferenceType _M_load

5.98.1 Detailed Description

template<typename _DifferenceTp>
struct __gnu_parallel::_Job< _DifferenceTp >

One __job for a certain thread.

5.98.2 Member Data Documentation

_M_first

```
template<typename _DifferenceTp>
volatile _DifferenceType __gnu_parallel::_Job< _DifferenceTp >::_M_first
```

First element.
Changed by owning and stealing thread. By stealing thread, always incremented.
Referenced by __gnu_parallel::__for_each_template_random_access_workstealing().

_M_last

```
template<typename _DifferenceTp>
volatile _DifferenceType __gnu_parallel::_Job< _DifferenceTp >::_M_last
```

Last element.
Changed by owning thread only.
Referenced by __gnu_parallel::__for_each_template_random_access_workstealing().

_M_load

```
template<typename _DifferenceTp>
volatile _DifferenceType __gnu_parallel::_Job< _DifferenceTp >::_M_load
```

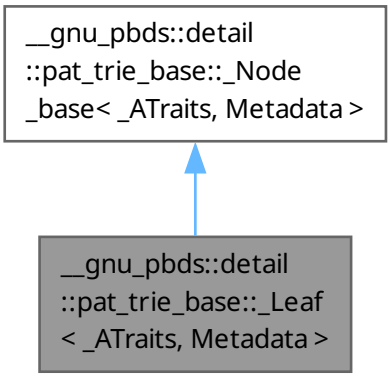
Number of elements, i.e. `_M_last-_M_first+1`.
Changed by owning thread only.
Referenced by __gnu_parallel::__for_each_template_random_access_workstealing().
The documentation for this struct was generated from the following file:

- workstealing.h

5.99 __gnu_pbds::detail::pat_trie_base::_Leaf< _ATraits, Metadata > Struct Template Reference

```
#include <pat_trie_base.hpp>
```

Inheritance diagram for __gnu_pbds::detail::pat_trie_base::_Leaf< _ATraits, Metadata >:



Public Types

- typedef _ATraits::const_iterator **a_const_iterator**
- typedef detail::rebind_traits< _Alloc, _ATraits >::const_pointer **a_const_pointer**
- typedef _ATraits **access_traits**
- typedef _Alloc **allocator_type**
- typedef _Node_base< _ATraits, Metadata > **base_type**
- typedef type_traits::const_reference **const_reference**
- typedef detail::rebind_traits< _Alloc, _Node_base >::pointer **node_pointer**
- typedef type_traits::reference **reference**
- typedef base_type::type_traits **type_traits**
- typedef type_traits::value_type **value_type**

Public Member Functions

- **_Leaf** (const_reference other)
- reference **value** ()
- const_reference **value** () const

Public Attributes

- node_pointer **m_p_parent**
- const node_type **m_type**

5.99.1 Detailed Description

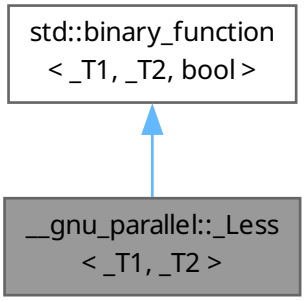
template<typename _ATraits, typename Metadata>
struct __gnu_pbds::detail::pat_trie_base::_Leaf< _ATraits, Metadata >

Leaf node for PATRICIA tree.
The documentation for this struct was generated from the following file:

- pat_trie_base.hpp

5.100 __gnu_parallel::_Less< _T1, _T2 > Struct Template Reference

`#include <base.h>`
Inheritance diagram for __gnu_parallel::_Less< _T1, _T2 >:



Public Types

- typedef _T1 first_argument_type
- typedef bool result_type
- typedef _T2 second_argument_type

Public Member Functions

- bool **operator()** (const _T1 &__t1, const _T2 &__t2) const
- bool **operator()** (const _T2 &__t2, const _T1 &__t1) const

5.100.1 Detailed Description

template<typename _T1, typename _T2>
struct __gnu_parallel::_Less< _T1, _T2 >

Similar to std::less, but allows two different types.

5.100.2 Member Typedef Documentation

first_argument_type

```
typedef _T1 std::binary_function< _T1, _T2, bool >::first_argument_type  [inherited]
```

`first_argument_type` is the type of the first argument

result_type

```
typedef bool std::binary_function< _T1, _T2, bool >::result_type  [inherited]
```

`result_type` is the return type

second_argument_type

```
typedef _T2 std::binary_function< _T1, _T2, bool >::second_argument_type  [inherited]
```

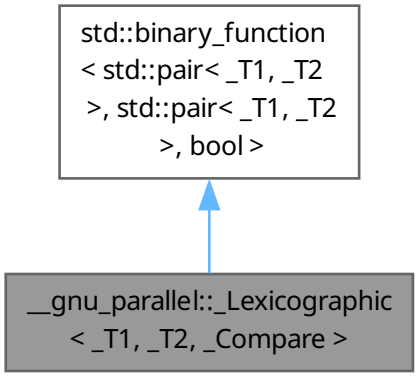
`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- base.h

5.101 __gnu_parallel::_Lexicographic< _T1, _T2, _Compare > Class Template Reference

```
#include <multiseq_selection.h>
```

Inheritance diagram for __gnu_parallel::_Lexicographic< _T1, _T2, _Compare >:



Public Types

- typedef std::pair< _T1, _T2 > first_argument_type
- typedef bool result_type
- typedef std::pair< _T1, _T2 > second_argument_type

Public Member Functions

- **_Lexicographic** (_Compare &__comp)
- bool **operator()** (const std::pair< _T1, _T2 > &__p1, const std::pair< _T1, _T2 > &__p2) const

5.101.1 Detailed Description

template<typename _T1, typename _T2, typename _Compare>
class __gnu_parallel::_Lexicographic< _T1, _T2, _Compare >

Compare __a pair of types lexicographically, ascending.

5.101.2 Member Typedef Documentation

first_argument_type

```
typedef std::pair< _T1, _T2 > std::binary_function< std::pair< _T1, _T2 >, std::pair< _T1, _T2
>, bool >::first_argument_type  [inherited]
```

`first_argument_type` is the type of the first argument

result_type

```
typedef bool std::binary_function< std::pair< _T1, _T2 >, std::pair< _T1, _T2 >, bool >::result↩
_type  [inherited]
```

`result_type` is the return type

second_argument_type

```
typedef std::pair< _T1, _T2 > std::binary_function< std::pair< _T1, _T2 >, std::pair< _T1, _T2 >, bool >::second_argument_type [inherited]
```

second_argument_type is the type of the second argument

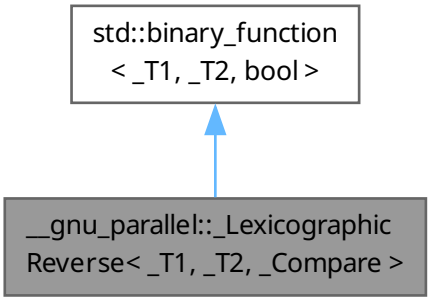
The documentation for this class was generated from the following file:

- multiseq_selection.h

5.102 __gnu_parallel::_LexicographicReverse< _T1, _T2, _Compare > Class Template Reference

```
#include <multiseq_selection.h>
```

Inheritance diagram for __gnu_parallel::_LexicographicReverse< _T1, _T2, _Compare >:



Public Types

- typedef _T1 first_argument_type
- typedef bool result_type
- typedef _T2 second_argument_type

Public Member Functions

- **_LexicographicReverse** (_Compare &__comp)
- bool **operator()** (const std::pair< _T1, _T2 > &__p1, const std::pair< _T1, _T2 > &__p2) const

5.102.1 Detailed Description

template<typename _T1, typename _T2, typename _Compare>
class __gnu_parallel::_LexicographicReverse< _T1, _T2, _Compare >

Compare __a pair of types lexicographically, descending.

5.102.2 Member Typedef Documentation

first_argument_type

```
typedef _T1 std::binary_function< _T1, _T2, bool >::first_argument_type [inherited]
```

first_argument_type is the type of the first argument

result_type

typedef bool std::binary_function< _T1, _T2, bool >::result_type [inherited]
result_type is the return type

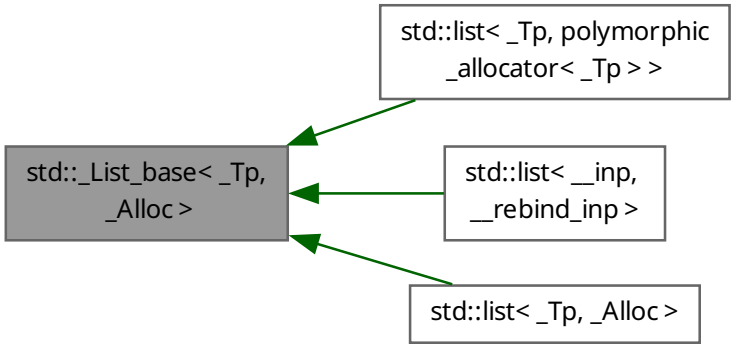
second_argument_type

typedef _T2 std::binary_function< _T1, _T2, bool >::second_argument_type [inherited]
second_argument_type is the type of the second argument
The documentation for this class was generated from the following file:

- multiseq_selection.h

5.103 std::_List_base< _Tp, _Alloc > Class Template Reference

#include <stl_list.h>
Inheritance diagram for std::_List_base< _Tp, _Alloc >:



Public Types

- typedef _Alloc **allocator_type**

Public Member Functions

- **_List_base** (_List_base &&)=default
- **_List_base** (_List_base &&__x, _Node_alloc_type &&__a)
- **_List_base** (_Node_alloc_type &&__a)
- **_List_base** (_Node_alloc_type &&__a, _List_base &&__x)
- **_List_base** (const _Node_alloc_type &__a) noexcept
- void **_M_clear** () noexcept
- size_t **_M_distance** (const __detail::_List_node_base *__first, const __detail::_List_node_base *__last) const
- const _Node_alloc_type & **_M_get_Node_allocator** () const noexcept
- _Node_alloc_type & **_M_get_Node_allocator** () noexcept
- void **_M_init** () noexcept
- void **_M_move_nodes** (_List_base &&__x)
- size_t **_M_node_count** () const

Static Public Member Functions

- static size_t **_S_distance** (const __detail::_List_node_base *__first, const __detail::_List_node_base *__last)

Protected Types

- typedef __gnu_cxx::__alloc_traits< _Node_alloc_type > **_Node_alloc_traits**
- typedef _Tp_alloc_traits::template rebind< typename_Node_traits::_Node >::other **_Node_alloc_type**
- using **_Node_ptr**
- typedef __list::_Node_traits< _Tp, typename _Tp_alloc_traits::pointer > **_Node_traits**
- typedef __gnu_cxx::__alloc_traits< _Tp_alloc_type > **_Tp_alloc_traits**
- typedef __gnu_cxx::__alloc_traits< _Alloc >::template rebind< _Tp >::other **_Tp_alloc_type**

Protected Member Functions

- void **_M_dec_size** (size_t __n)
- void **_M_destroy_node** (_Node_ptr __p)
- _Node_alloc_traits::pointer **_M_get_node** ()
- size_t **_M_get_size** () const
- void **_M_inc_size** (size_t __n)
- void **_M_put_node** (_Node_ptr __p) noexcept
- void **_M_set_size** (size_t __n)

Protected Attributes

- _List_impl **_M_impl**

5.103.1 Detailed Description

template<typename _Tp, typename _Alloc>
class std::_List_base< _Tp, _Alloc >

See bits/stl_deque.h's _Deque_base for an explanation.
The documentation for this class was generated from the following files:

- stl_list.h
- list.tcc

5.104 std::_List_const_iterator< _Tp > Struct Template Reference

`#include <stl_list.h>`

Public Types

- typedef const _List_node< _Tp > **_Node**
- typedef ptrdiff_t **difference_type**
- typedef _List_iterator< _Tp > **iterator**
- typedef std::bidirectional_iterator_tag **iterator_category**
- typedef const _Tp * **pointer**
- typedef const _Tp & **reference**
- typedef _Tp **value_type**

Public Member Functions

- **_List_const_iterator** (const __detail::_List_node_base *__x) noexcept
- **_List_const_iterator** (const iterator &__x) noexcept
- iterator **_M_const_cast** () const noexcept
- reference **operator*** () const noexcept
- _List_const_iterator & **operator++** () noexcept
- _List_const_iterator **operator++** (int) noexcept
- _List_const_iterator & **operator--** () noexcept
- _List_const_iterator **operator--** (int) noexcept
- pointer **operator->** () const noexcept

Public Attributes

- const __detail::_List_node_base * **_M_node**

Friends

- bool **operator==** (const _List_const_iterator &__x, const _List_const_iterator &__y) noexcept

5.104.1 Detailed Description

template<typename _Tp>
struct std::_List_const_iterator< _Tp >

A list::const_iterator.
All the functions are op overloads.
The documentation for this struct was generated from the following files:

- stl_iterator_base_funcs.h
- stl_list.h

5.105 std::_List_iterator< _Tp > Struct Template Reference

`#include <stl_list.h>`

Public Types

- typedef _List_node< _Tp > **_Node**
- typedef ptrdiff_t **difference_type**
- typedef std::bidirectional_iterator_tag **iterator_category**
- typedef _Tp * **pointer**
- typedef _Tp & **reference**
- typedef _Tp **value_type**

Public Member Functions

- **_List_iterator** (__detail::_List_node_base *__x) noexcept
- _List_iterator **_M_const_cast** () const noexcept
- reference **operator*** () const noexcept
- _List_iterator & **operator++** () noexcept
- _List_iterator **operator++** (int) noexcept
- _List_iterator & **operator--** () noexcept
- _List_iterator **operator--** (int) noexcept
- pointer **operator->** () const noexcept

Public Attributes

- __detail::_List_node_base * **_M_node**

Friends

- bool **operator==** (const _List_iterator &__x, const _List_iterator &__y) noexcept

5.105.1 Detailed Description

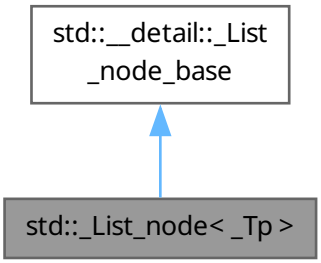
template<typename _Tp>
struct std::_List_iterator< _Tp >

A list::iterator.
All the functions are op overloads.
The documentation for this struct was generated from the following files:

- stl_iterator_base_funcs.h
- stl_list.h

5.106 std::_List_node< _Tp > Struct Template Reference

`#include <stl_list.h>`
Inheritance diagram for std::_List_node< _Tp >:



Public Types

- typedef _List_node_base * **_Base_ptr**
- typedef _List_node * **_Node_ptr**

Public Member Functions

- _List_node_base * **_M_base** ()
- const _List_node_base * **_M_base** () const
- void **_M_hook** (_List_node_base *const __position) noexcept
- _Node_ptr **_M_node_ptr** ()
- void **_M_reverse** () noexcept
- void **_M_transfer** (_List_node_base *const __first, _List_node_base *const __last) noexcept
- void **_M_unhook** () noexcept

- _Tp * **_M_valptr** ()
- _Tp const * **_M_valptr** () const

Static Public Member Functions

- static void **swap** (_List_node_base &__x, _List_node_base &__y) noexcept

Public Attributes

- _List_node_base * **_M_next**
- _List_node_base * **_M_prev**
- __gnu_cxx::__aligned_membuf< _Tp > **_M_storage**

5.106.1 Detailed Description

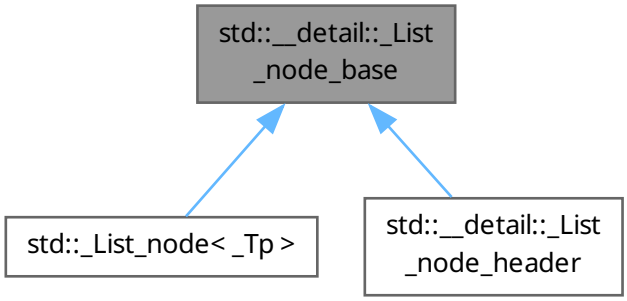
template<typename _Tp>
struct std::_List_node< _Tp >

An actual node in the list.
The documentation for this struct was generated from the following file:

- stl_list.h

5.107 std::__detail::_List_node_base Struct Reference

`#include <stl_list.h>`
Inheritance diagram for std::__detail::_List_node_base:



Public Types

- typedef _List_node_base * **_Base_ptr**

Public Member Functions

- _List_node_base * **_M_base** ()
- const _List_node_base * **_M_base** () const
- void **_M_hook** (_List_node_base *const __position) noexcept

- void **_M_reverse** () noexcept
- void **_M_transfer** (_List_node_base *const __first, _List_node_base *const __last) noexcept
- void **_M_unhook** () noexcept

Static Public Member Functions

- static void **swap** (_List_node_base &__x, _List_node_base &__y) noexcept

Public Attributes

- _List_node_base * **_M_next**
- _List_node_base * **_M_prev**

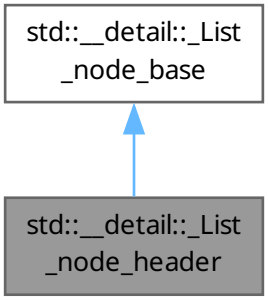
5.107.1 Detailed Description

Common part of a node in the list.
The documentation for this struct was generated from the following file:

- stl_list.h

5.108 std::__detail::_List_node_header Struct Reference

`#include <stl_list.h>`
Inheritance diagram for std::__detail::_List_node_header:



Public Types

- typedef _List_node_base * **_Base_ptr**

Public Member Functions

- **_List_node_header** (_List_node_header &&__x) noexcept
- _List_node_base * **_M_base** ()
- const _List_node_base * **_M_base** () const
- void **_M_hook** (_List_node_base *const __position) noexcept
- void **_M_init** () noexcept
- void **_M_move_nodes** (_List_node_header &&__x)

- void **_M_reverse** () noexcept
- void **_M_transfer** (_List_node_base *const __first, _List_node_base *const __last) noexcept
- void **_M_unhook** () noexcept

Static Public Member Functions

- static void **swap** (_List_node_base &__x, _List_node_base &__y) noexcept

Public Attributes

- _List_node_base * **_M_next**
- _List_node_base * **_M_prev**
- size_t **_M_size**

5.108.1 Detailed Description

The list node header.
The documentation for this struct was generated from the following file:

- stl_list.h

5.109 __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser Struct Reference

`#include <losertree.h>`

Public Attributes

- _Tp _M_key
- int _M_source
- bool _M_sup

5.109.1 Detailed Description

template<typename _Tp, typename _Compare>
struct __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser

Internal representation of a _LoserTree element.

5.109.2 Member Data Documentation

_M_key

```
template<typename _Tp, typename _Compare>
_Tp __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_key
```

_M_key of the element in the _LoserTree.

_M_source

```
template<typename _Tp, typename _Compare>
int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source
```

__index of the __source __sequence.

_M_sup

```
template<typename _Tp, typename _Compare>
bool __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_sup
```

flag, true iff this is a "maximum" __sentinel.
The documentation for this struct was generated from the following file:

- losertree.h

5.110 __gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >::_Loser Struct Reference

```
#include <losertree.h>
```

Public Attributes

- const _Tp * **_M_keyp**
- int **_M_source**
- bool **_M_sup**

5.110.1 Detailed Description

template<typename _Tp, typename _Compare>
struct __gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >::_Loser

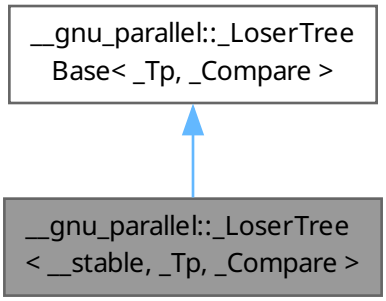
Internal representation of _LoserTree __elements.
The documentation for this struct was generated from the following file:

- losertree.h

5.111 __gnu_parallel::_LoserTree< __stable, _Tp, _Compare > Class Template Reference

```
#include <losertree.h>
```

Inheritance diagram for __gnu_parallel::_LoserTree< __stable, _Tp, _Compare >:



Public Member Functions

- **_LoserTree** (unsigned int __k, _Compare __comp)

- void __delete_min_insert (_Tp __key, bool __sup)
- int __get_min_source ()
- void **__init** ()
- unsigned int **__init_winner** (unsigned int __root)
- void __insert_start (const _Tp &__key, int __source, bool __sup)

Protected Attributes

- unsigned int **_M_ik**
- unsigned int _M_log_k
- unsigned int **_M_offset**

5.111.1 Detailed Description

template<bool __stable, typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTree< __stable, _Tp, _Compare >

Stable _LoserTree variant.
Provides the stable implementations of insert_start, __init_winner, __init and __delete_min_insert.
Unstable variant is done using partial specialisation below.

5.111.2 Member Function Documentation

__delete_min_insert()

```
template<bool __stable, typename _Tp, typename _Compare>
void __gnu_parallel::_LoserTree< __stable, _Tp, _Compare >::__delete_min_insert (
            _Tp __key,
            bool __sup)  [inline]
```

Delete the smallest element and insert a new element from the previously smallest element's sequence.
This implementation is stable.

__get_min_source()

```
template<typename _Tp, typename _Compare>
int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__get_min_source ()  [inline], [inherited]
```

Returns

the index of the sequence with the smallest element.

References _M_losers.

__insert_start()

```
template<typename _Tp, typename _Compare>
void __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start (
            const _Tp & __key,
            int __source,
            bool __sup)  [inline], [inherited]
```

Initializes the sequence "_M_source" with the element "__key".

Parameters

__key	the element to insert
__source	__index of the __source __sequence

__sup	flag that determines whether the value to insert is an explicit __supremum.

References _M_first_insert, and _M_losers.

5.111.3 Member Data Documentation

_M_log_k

```
template<typename _Tp, typename _Compare>
unsigned int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_log_k  [protected], [inherited]
```

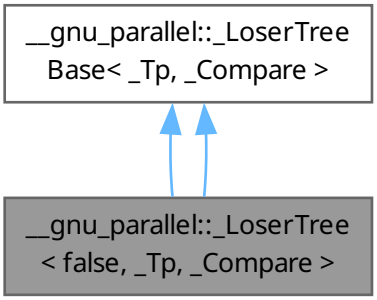
log_2{_M_k}
Referenced by _LoserTreeBase().
The documentation for this class was generated from the following file:

- losertree.h

5.112 __gnu_parallel::_LoserTree< false, _Tp, _Compare > Class Template Reference

```
#include <losertree.h>
```

Inheritance diagram for __gnu_parallel::_LoserTree< false, _Tp, _Compare >:



Public Member Functions

- **_LoserTree** (unsigned int __k, _Compare __comp)
- **_LoserTree** (unsigned int __k, _Compare __comp)
- void __delete_min_insert (_Tp __key, bool __sup)
- void __delete_min_insert (_Tp __key, bool __sup)
- int __get_min_source ()
- void **__init** ()
- void **__init** ()
- unsigned int **__init_winner** (unsigned int __root)
- unsigned int __init_winner (unsigned int __root)
- void __insert_start (const _Tp &__key, int __source, bool __sup)

Protected Attributes

- unsigned int **_M_ik**
- unsigned int _M_log_k
- unsigned int **_M_offset**

5.112.1 Detailed Description

template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTree< false, _Tp, _Compare >

Unstable _LoserTree variant.
Stability (non-stable here) is selected with partial specialization.

5.112.2 Member Function Documentation

__delete_min_insert() [1/2]

```
void __gnu_parallel::_LoserTree< __stable, _Tp, _Compare >::__delete_min_insert (
            _Tp __key,
            bool __sup)  [inline]
```

Delete the smallest element and insert a new element from the previously smallest element's sequence.
This implementation is stable.

__delete_min_insert() [2/2]

```
template<typename _Tp, typename _Compare>
void __gnu_parallel::_LoserTree< false, _Tp, _Compare >::__delete_min_insert (
            _Tp __key,
            bool __sup)  [inline]
```

Delete the _M_key smallest element and insert the element __key instead.

Parameters

__key	the _M_key to insert
__sup	true iff __key is an explicitly marked supremum

__get_min_source()

```
int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__get_min_source ()  [inline]
```

Returns

the index of the sequence with the smallest element.

__init_winner()

```
template<typename _Tp, typename _Compare>
unsigned int __gnu_parallel::_LoserTree< false, _Tp, _Compare >::__init_winner (
            unsigned int __root)  [inline]
```

Computes the winner of the competition at position "__root".
Called recursively (starting at 0) to build the initial tree.

Parameters

__root	__index of the "game" to start.

References __init_winner().
Referenced by __init_winner().

__insert_start()

```
void __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start (
            const _Tp & __key,
            int __source,
            bool __sup)  [inline]
```

Initializes the sequence "_M_source" with the element "__key".

Parameters

__key	the element to insert
__source	__index of the __source __sequence
__sup	flag that determines whether the value to insert is an explicit __supremum.

5.112.3 Member Data Documentation

_M_log_k

```
unsigned int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_log_k  [protected]
```

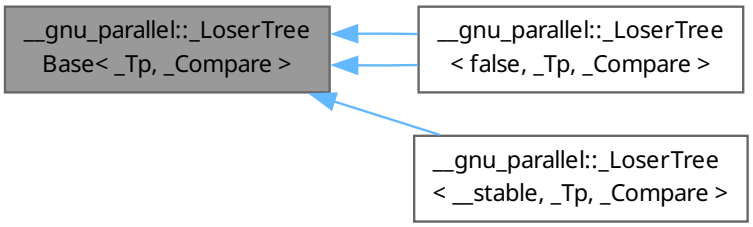
log_2{_M_k}
The documentation for this class was generated from the following file:

- losertree.h

5.113 __gnu_parallel::_LoserTreeBase< _Tp, _Compare > Class Template Reference

```
#include <losertree.h>
```

Inheritance diagram for __gnu_parallel::_LoserTreeBase< _Tp, _Compare >:



Classes

- struct _Loser

Public Member Functions

- _LoserTreeBase (unsigned int __k, _Compare __comp)
- ~_LoserTreeBase ()
- int __get_min_source ()
- void __insert_start (const _Tp &__key, int __source, bool __sup)

Protected Attributes

- _Compare _M_comp
- bool _M_first_insert
- unsigned int **_M_ik**
- unsigned int **_M_k**
- unsigned int _M_log_k
- _Loser * _M_losers
- unsigned int **_M_offset**

5.113.1 Detailed Description

template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreeBase< _Tp, _Compare >

Guarded loser/tournament tree.
The smallest element is at the top.
Guarding is done explicitly through one flag _M_sup per element, inf is not needed due to a better initialization routine.
This is a well-performing variant.

Parameters

_Tp	the element type
_Compare	the comparator to use, defaults to std::less<_Tp>

5.113.2 Constructor & Destructor Documentation

_LoserTreeBase()

```
template<typename _Tp, typename _Compare>
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase (
            unsigned int __k,
            _Compare __comp)  [inline]
```

The constructor.

Parameters

__k	The number of sequences to merge.
__comp	The comparator to use.

References __gnu_parallel::__rd_log2(), _M_comp, _M_first_insert, _M_log_k, and _M_losers.

~_LoserTreeBase()

```
template<typename _Tp, typename _Compare>
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::~_LoserTreeBase ()  [inline]
```

The destructor.
References _M_losers.

5.113.3 Member Function Documentation

__get_min_source()

```
template<typename _Tp, typename _Compare>
int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__get_min_source ()  [inline]
```

Returns

the index of the sequence with the smallest element.

References _M_losers.

__insert_start()

```
template<typename _Tp, typename _Compare>
void __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start (
            const _Tp & __key,
            int __source,
            bool __sup)  [inline]
```

Initializes the sequence "_M_source" with the element "__key".

Parameters

__key	the element to insert
__source	__index of the __source __sequence
__sup	flag that determines whether the value to insert is an explicit __supremum.

References _M_first_insert, and _M_losers.

5.113.4 Member Data Documentation

_M_comp

```
template<typename _Tp, typename _Compare>
_Compare __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_comp  [protected]
```

_Compare to use.
Referenced by _LoserTreeBase().

_M_first_insert

```
template<typename _Tp, typename _Compare>
bool __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_first_insert  [protected]
```

State flag that determines whether the _LoserTree is empty.
Only used for building the _LoserTree.
Referenced by _LoserTreeBase(), and __insert_start().

_M_log_k

```
template<typename _Tp, typename _Compare>
unsigned int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_log_k  [protected]
```

log_2{_M_k}
Referenced by _LoserTreeBase().

_M_losers

```
template<typename _Tp, typename _Compare>
_Loser* __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers  [protected]
```

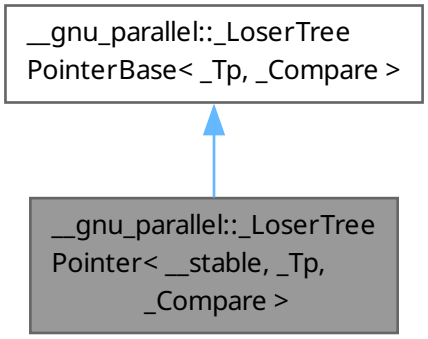
_LoserTree __elements.
Referenced by _LoserTreeBase(), ~_LoserTreeBase(), __get_min_source(), and __insert_start().
The documentation for this class was generated from the following file:

- losertree.h

5.114 __gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare > Class Template Reference

```
#include <losertree.h>
```

Inheritance diagram for __gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >:



Public Member Functions

- **_LoserTreePointer** (unsigned int __k, _Compare __comp=std::less< _Tp >())
- void **__delete_min_insert** (const _Tp &__key, bool __sup)
- int **__get_min_source** ()
- void **__init** ()
- unsigned int **__init_winner** (unsigned int __root)
- void **__insert_start** (const _Tp &__key, int __source, bool __sup)

Protected Attributes

- unsigned int **_M_ik**
- unsigned int **_M_offset**

5.114.1 Detailed Description

template<bool __stable, typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >

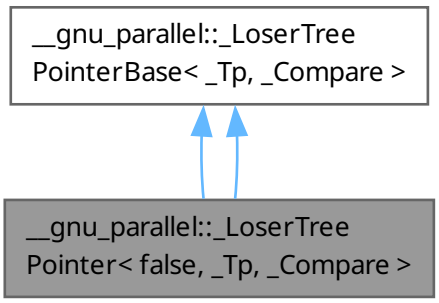
Stable _LoserTree implementation.

The unstable variant is implemented using partial instantiation below.
The documentation for this class was generated from the following file:

- losertree.h

5.115 __gnu_parallel::_LoserTreePointer< false, _Tp, _Compare > Class Template Reference

`#include <losertree.h>`
Inheritance diagram for __gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >:



Public Member Functions

- **_LoserTreePointer** (unsigned int __k, _Compare __comp=std::less< _Tp >())
- **_LoserTreePointer** (unsigned int __k, _Compare __comp=std::less< _Tp >())
- void **__delete_min_insert** (const _Tp &__key, bool __sup)
- void **__delete_min_insert** (const _Tp &__key, bool __sup)
- int **__get_min_source** ()
- void **__init** ()
- void **__init** ()
- unsigned int **__init_winner** (unsigned int __root)
- unsigned int **__init_winner** (unsigned int __root)
- void **__insert_start** (const _Tp &__key, int __source, bool __sup)

Protected Attributes

- unsigned int **_M_ik**
- unsigned int **_M_offset**

5.115.1 Detailed Description

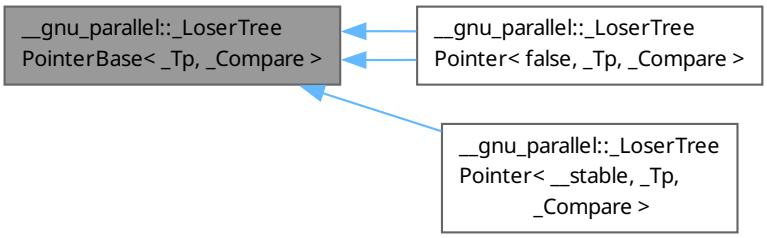
template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >

Unstable _LoserTree implementation.
The stable variant is above.
The documentation for this class was generated from the following file:

- losertree.h

5.116 __gnu_parallel::_LoserTreePointerBase< _Tp, _Compare > Class Template Reference

`#include <losertree.h>`
Inheritance diagram for __gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >:



Classes

- struct _Loser

Public Member Functions

- **_LoserTreePointerBase** (unsigned int __k, _Compare __comp=std::less< _Tp >())
- int **__get_min_source** ()
- void **__insert_start** (const _Tp &__key, int __source, bool __sup)

Protected Attributes

- _Compare **_M_comp**
- unsigned int **_M_ik**
- unsigned int **_M_k**
- _Loser * **_M_losers**
- unsigned int **_M_offset**

5.116.1 Detailed Description

template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >

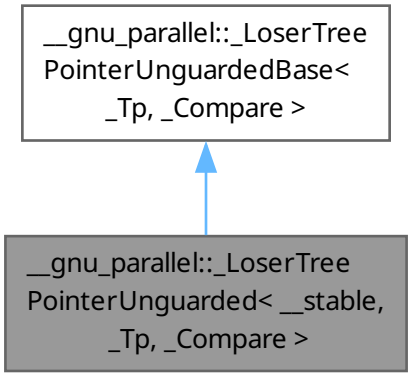
Base class of _Loser Tree implementation using pointers.
The documentation for this class was generated from the following file:

- losertree.h

5.117 __gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare > Class Template Reference

`#include <losertree.h>`

Inheritance diagram for __gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >:



Public Member Functions

- **_LoserTreePointerUnguarded** (unsigned int __k, const _Tp &__sentinel, _Compare __comp=std::less< _Tp >())
- void **__delete_min_insert** (const _Tp &__key, bool __sup)
- int **__get_min_source** ()
- void **__init** ()
- unsigned int **__init_winner** (unsigned int __root)
- void **__insert_start** (const _Tp &__key, int __source, bool)

Protected Attributes

- unsigned int **_M_ik**
- unsigned int **_M_offset**

5.117.1 Detailed Description

template<bool __stable, typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >

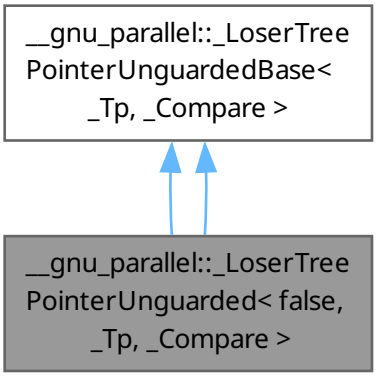
Stable unguarded _LoserTree variant storing pointers.
Unstable variant is implemented below using partial specialization.
The documentation for this class was generated from the following file:

- losertree.h

5.118 __gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare > Class Template Reference

`#include <losertree.h>`

Inheritance diagram for __gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >:



Public Member Functions

- **_LoserTreePointerUnguarded** (unsigned int __k, const _Tp &__sentinel, _Compare __comp=std::less< _Tp >())
- **_LoserTreePointerUnguarded** (unsigned int __k, const _Tp &__sentinel, _Compare __comp=std::less< _Tp >())
- void **__delete_min_insert** (const _Tp &__key, bool __sup)
- void **__delete_min_insert** (const _Tp &__key, bool __sup)
- int **__get_min_source** ()
- void **__init** ()
- void **__init** ()
- unsigned int **__init_winner** (unsigned int __root)
- unsigned int **__init_winner** (unsigned int __root)
- void **__insert_start** (const _Tp &__key, int __source, bool)

Protected Attributes

- unsigned int **_M_ik**
- unsigned int **_M_offset**

5.118.1 Detailed Description

template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >

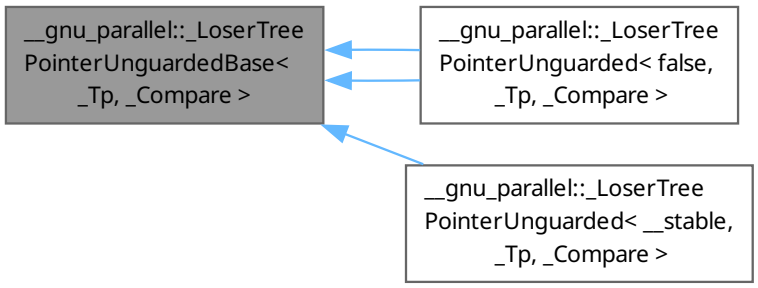
Unstable unguarded _LoserTree variant storing pointers.
Stable variant is above.
The documentation for this class was generated from the following file:

- losertree.h

5.119 __gnu_parallel::_LoserTreePointerUnguardedBase< _Tp, _Compare > Class Template Reference

`#include <losertree.h>`

Inheritance diagram for __gnu_parallel::_LoserTreePointerUnguardedBase< _Tp, _Compare >:



Public Member Functions

- **_LoserTreePointerUnguardedBase** (unsigned int __k, const _Tp &__sentinel, _Compare __comp=std::less< _Tp >())
- int **__get_min_source** ()
- void **__insert_start** (const _Tp &__key, int __source, bool)

Protected Attributes

- _Compare **_M_comp**
- unsigned int **_M_ik**
- unsigned int **_M_k**
- _Loser * **_M_losers**
- unsigned int **_M_offset**

5.119.1 Detailed Description

template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreePointerUnguardedBase< _Tp, _Compare >

Unguarded loser tree, keeping only pointers to the elements in the tree structure.
No guarding is done, therefore not a single input sequence must run empty. This is a very fast variant.
The documentation for this class was generated from the following file:

- losertree.h

5.120 __gnu_parallel::_LoserTreeTraits< _Tp > Struct Template Reference

`#include <multiway_merge.h>`

Static Public Attributes

- static const bool _M_use_pointer

5.120.1 Detailed Description

template<typename _Tp>
struct __gnu_parallel::_LoserTreeTraits< _Tp >

Traits for determining whether the loser tree should use pointers or copies.
The field "_M_use_pointer" is used to determine whether to use pointers in he loser trees or whether to copy the values into the loser tree.
The default behavior is to use pointers if the data type is 4 times as big as the pointer to it.
Specialize for your data type to customize the behavior.
Example:
template<> struct _LoserTreeTraits<int> { static const bool _M_use_pointer = false; };
template<> struct _LoserTreeTraits<heavyweight_type> { static const bool _M_use_pointer = true; };

Parameters

_Tp	type to give the loser tree traits for.

5.120.2 Member Data Documentation

_M_use_pointer

```
template<typename _Tp>
const bool __gnu_parallel::_LoserTreeTraits< _Tp >::_M_use_pointer  [static]
```

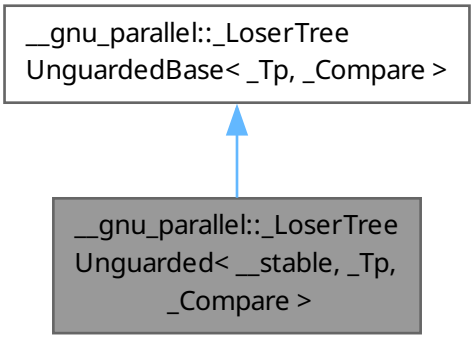
True iff to use pointers instead of values in loser trees.
The default behavior is to use pointers if the data type is four times as big as the pointer to it.
The documentation for this struct was generated from the following file:

- multiway_merge.h

5.121 __gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare > Class Template Reference

```
#include <losertree.h>
```

Inheritance diagram for __gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >:



Public Member Functions

- **_LoserTreeUnguarded** (unsigned int __k, const _Tp &__sentinel, _Compare __comp=std::less< _Tp >())
- void **__delete_min_insert** (_Tp __key, bool)
- int **__get_min_source** ()
- void **__init** ()
- unsigned int **__init_winner** (unsigned int __root)
- void **__insert_start** (const _Tp &__key, int __source, bool)

Protected Attributes

- unsigned int **_M_ik**
- unsigned int **_M_offset**

5.121.1 Detailed Description

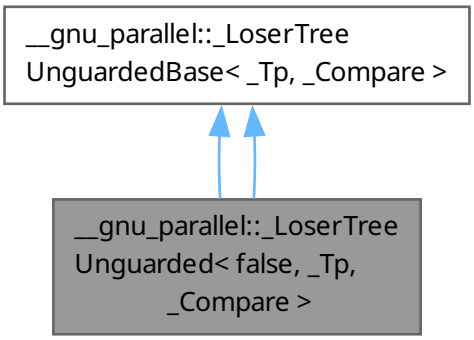
template<bool __stable, typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >

Stable implementation of unguarded _LoserTree.
Unstable variant is selected below with partial specialization.
The documentation for this class was generated from the following file:

- losertree.h

5.122 __gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare > Class Template Reference

`#include <losertree.h>`
Inheritance diagram for __gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >:



Public Member Functions

- **_LoserTreeUnguarded** (unsigned int __k, const _Tp &__sentinel, _Compare __comp=std::less< _Tp >())
- **_LoserTreeUnguarded** (unsigned int __k, const _Tp &__sentinel, _Compare __comp=std::less< _Tp >())
- void **__delete_min_insert** (_Tp __key, bool)

- void **__delete_min_insert** (_Tp __key, bool)
- int **__get_min_source** ()
- void **__init** ()
- void **__init** ()
- unsigned int **__init_winner** (unsigned int __root)
- unsigned int **__init_winner** (unsigned int __root)
- void **__insert_start** (const _Tp &__key, int __source, bool)

Protected Attributes

- unsigned int **_M_ik**
- unsigned int **_M_offset**

5.122.1 Detailed Description

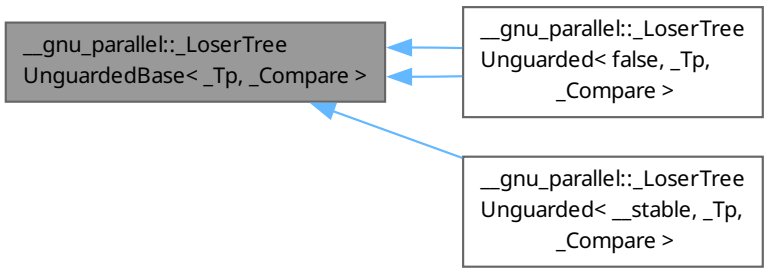
template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >

Non-Stable implementation of unguarded _LoserTree.
Stable implementation is above.
The documentation for this class was generated from the following file:

- losertree.h

5.123 __gnu_parallel::_LoserTreeUnguardedBase< _Tp, _Compare > Class Template Reference

`#include <losertree.h>`
Inheritance diagram for __gnu_parallel::_LoserTreeUnguardedBase< _Tp, _Compare >:



Public Member Functions

- **_LoserTreeUnguardedBase** (unsigned int __k, const _Tp &__sentinel, _Compare __comp=std::less< _Tp >())
- int **__get_min_source** ()
- void **__insert_start** (const _Tp &__key, int __source, bool)

Protected Attributes

- _Compare **_M_comp**
- unsigned int **_M_ik**
- unsigned int **_M_k**
- _Loser * **_M_losers**
- unsigned int **_M_offset**

5.123.1 Detailed Description

template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreeUnguardedBase< _Tp, _Compare >

Base class for unguarded _LoserTree implementation.
The whole element is copied into the tree structure.
No guarding is done, therefore not a single input sequence must run empty. Unused __sequence heads are marked with a sentinel which is > all elements that are to be merged.
This is a very fast variant.
The documentation for this class was generated from the following file:

- losertree.h

5.124 __gnu_pbds::detail::pat_trie_base::_Metadata< Metadata, _Alloc > Struct Template Reference

`#include <pat_trie_base.hpp>`

Public Types

- typedef _Alloc **allocator_type**
- typedef detail::rebind_traits< _Alloc, Metadata >::const_reference **const_reference**
- typedef Metadata **metadata_type**

Public Member Functions

- const_reference **get_metadata** () const

Public Attributes

- metadata_type **m_metadata**

5.124.1 Detailed Description

template<typename Metadata, typename _Alloc>
struct __gnu_pbds::detail::pat_trie_base::_Metadata< Metadata, _Alloc >

Metadata base primary template.
The documentation for this struct was generated from the following file:

- pat_trie_base.hpp

5.125 __gnu_pbds::detail::pat_trie_base::_Metadata< null_type, _Alloc > Struct Template Reference

`#include <pat_trie_base.hpp>`

Public Types

- typedef _Alloc **allocator_type**
- typedef _Alloc **allocator_type**
- typedef detail::rebind_traits< _Alloc, null_type >::const_reference **const_reference**
- typedef null_type **metadata_type**
- typedef null_type **metadata_type**

Public Member Functions

- const_reference **get_metadata** () const

Public Attributes

- metadata_type **m_metadata**

5.125.1 Detailed Description

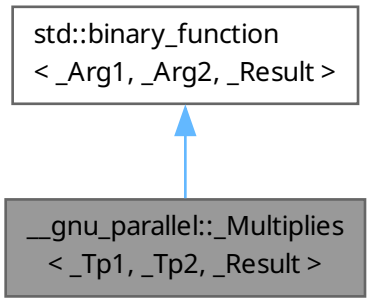
template<typename _Alloc>
struct __gnu_pbds::detail::pat_trie_base::_Metadata< null_type, _Alloc >

Specialization for null metadata.
The documentation for this struct was generated from the following file:

- pat_trie_base.hpp

5.126 __gnu_parallel::_Multiplies< _Tp1, _Tp2, _Result > Struct Template Reference

`#include <base.h>`
Inheritance diagram for __gnu_parallel::_Multiplies< _Tp1, _Tp2, _Result >:



Public Types

- typedef _Arg1 first_argument_type
- typedef _Result result_type
- typedef _Arg2 second_argument_type

Public Member Functions

- _Result **operator()** (const _Tp1 &__x, const _Tp2 &__y) const

5.126.1 Detailed Description

template<typename _Tp1, typename _Tp2, typename _Result = __typeof__(*static_cast<_Tp1*>(0) * *static_cast<_Tp2*>(0))>
struct __gnu_parallel::_Multiplies< _Tp1, _Tp2, _Result >

Similar to std::multiplies, but allows two different types.

5.126.2 Member Typedef Documentation

first_argument_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result >::first_argument_type  [inherited]
```

`first_argument_type` is the type of the first argument

result_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Result std::binary_function< _Arg1, _Arg2, _Result >::result_type  [inherited]
```

`result_type` is the return type

second_argument_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result >::second_argument_type  [inherited]
```

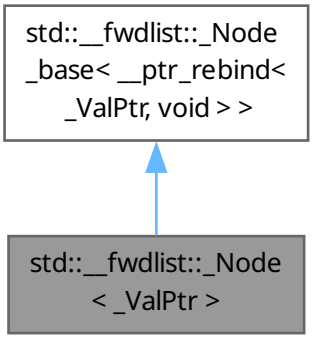
`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- base.h

5.127 std::__fwdlist::_Node< _ValPtr > Struct Template Reference

`#include <forward_list.h>`

Inheritance diagram for std::__fwdlist::_Node< _ValPtr >:



Public Types

- using **_Base_ptr**
- using **_Node_ptr**
- using **value_type**

Public Member Functions

- **_Node** (_Node &&)=delete
- _Base_ptr **_M_base_ptr** () const
- _Node_ptr **_M_node_ptr** ()
- void **_M_reverse_after** () noexcept
- _Base_ptr **_M_transfer_after** (_Base_ptr __begin, _Base_ptr __end) noexcept
- const value_type * **_M_valptr** () const noexcept
- value_type * **_M_valptr** () noexcept

Public Attributes

- _Base_ptr **_M_next**
- _Uninit_storage **_M_u**

5.127.1 Detailed Description

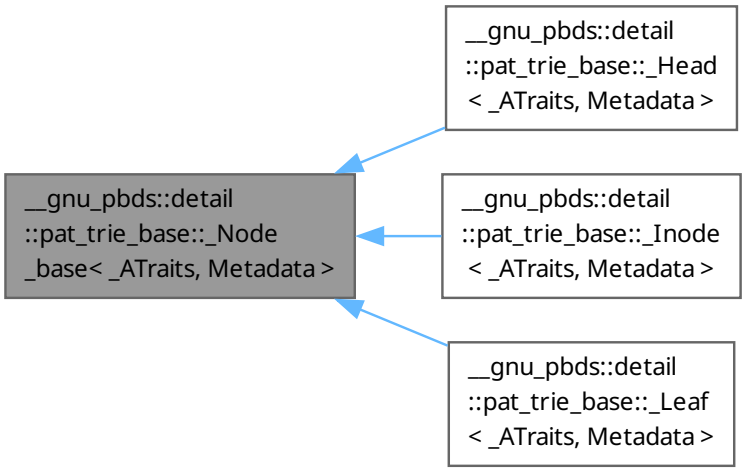
template<typename _ValPtr>
struct std::__fwdlist::_Node< _ValPtr >

A helper node class for forward_list.
The documentation for this struct was generated from the following file:

- forward_list.h

5.128 __gnu_pbds::detail::pat_trie_base::_Node_base< _ATraits, Metadata > Struct Template Reference

`#include <pat_trie_base.hpp>`
Inheritance diagram for __gnu_pbds::detail::pat_trie_base::_Node_base< _ATraits, Metadata >:



Public Types

- typedef _ATraits::const_iterator **a_const_iterator**
- typedef detail::rebind_traits< _Alloc, _ATraits >::const_pointer **a_const_pointer**
- typedef _ATraits **access_traits**
- typedef _Alloc **allocator_type**
- typedef detail::rebind_traits< _Alloc, _Node_base >::pointer **node_pointer**
- typedef _ATraits::type_traits **type_traits**

Public Member Functions

- **_Node_base** (node_type type)

Public Attributes

- node_pointer **m_p_parent**
- const node_type **m_type**

5.128.1 Detailed Description

template<typename _ATraits, typename Metadata>
struct __gnu_pbds::detail::pat_trie_base::_Node_base< _ATraits, Metadata >

Node base.
The documentation for this struct was generated from the following file:

- pat_trie_base.hpp

5.129 std::__fwdlist::_Node_base< _VoidPtr > Struct Template Reference

`#include <forward_list.h>`

Public Types

- using **_Base_ptr**

Public Member Functions

- **_Node_base** (_Node_base &&__x) noexcept
- **_Node_base** (const _Node_base &)=delete
- _Base_ptr **_M_base_ptr** () const
- void **_M_reverse_after** () noexcept
- _Base_ptr **_M_transfer_after** (_Base_ptr __begin, _Base_ptr __end) noexcept
- _Node_base & **operator=** (_Node_base &&__x) noexcept
- _Node_base & **operator=** (const _Node_base &)=delete

Public Attributes

- _Base_ptr **_M_next**

5.129.1 Detailed Description

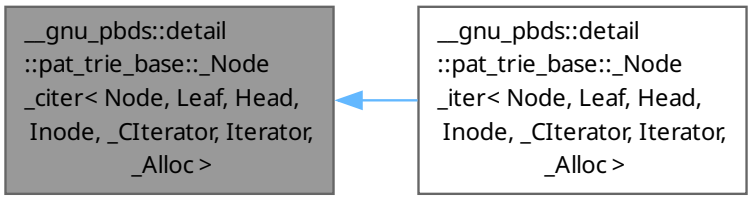
template<typename _VoidPtr>
struct std::__fwdlist::_Node_base< _VoidPtr >

The node-base type for allocators that use fancy pointers.
The documentation for this struct was generated from the following file:

- forward_list.h

5.130 __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > Class Template Reference

`#include <pat_trie_base.hpp>`
Inheritance diagram for __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >:



Public Types

- typedef value_type **const_reference**
- typedef trivial_iterator_difference_type **difference_type**
- typedef trivial_iterator_tag **iterator_category**
- typedef rebind_traits< _Alloc, metadata_type >::const_reference metadata_const_reference
- typedef Node::metadata_type metadata_type
- typedef value_type **reference**
- typedef _Alloc::size_type **size_type**
- typedef _CIterator **value_type**

Public Member Functions

- **_Node_citer** (node_pointer p_nd=0, a_const_pointer p_traits=0)
- _Node_citer get_child (size_type i) const
- metadata_const_reference get_metadata () const
- size_type num_children () const
- bool operator!= (const _Node_citer &other) const
- const_reference operator* () const
- bool operator== (const _Node_citer &other) const
- std::pair< a_const_iterator, a_const_iterator > valid_prefix () const

Public Attributes

- node_pointer **m_p_nd**
- a_const_pointer **m_p_traits**

Protected Types

- typedef Node::a_const_iterator **a_const_iterator**
- typedef Node::a_const_pointer **a_const_pointer**
- typedef rebind_traits< _Alloc, Inode >::const_pointer **inode_const_pointer**
- typedef rebind_traits< _Alloc, Inode >::pointer **inode_pointer**
- typedef rebind_traits< _Alloc, Leaf >::const_pointer **leaf_const_pointer**
- typedef rebind_traits< _Alloc, Leaf >::pointer **leaf_pointer**
- typedef rebind_traits< _Alloc, Node >::pointer **node_pointer**

5.130.1 Detailed Description

template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename Iterator, typename _Alloc>
class __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >

Node const iterator.

5.130.2 Member Typedef Documentation

metadata_const_reference

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
typedef rebind_traits<_Alloc,metadata_type>::const_reference __gnu_pbds::detail::pat_trie_base::_Node_citer<
Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::metadata_const_reference
```

Const metadata reference type.

metadata_type

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
typedef Node::metadata_type __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head,
Inode, _CIterator, Iterator, _Alloc >::metadata_type
```

Metadata type.

5.130.3 Member Function Documentation

get_child()

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
_Node_citer __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator,
Iterator, _Alloc >::get_child (
            size_type i) const  [inline]
```

Returns a __const node __iterator to the corresponding node's i-th child.

get_metadata()

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
metadata_const_reference __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode,
_CIterator, Iterator, _Alloc >::get_metadata () const  [inline]
```

Metadata access.

num_children()

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
size_type __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator,
Iterator, _Alloc >::num_children () const  [inline]
```

Returns the number of children in the corresponding node.

Referenced by __gnu_pbds::detail::pat_trie_base::_Node_citer< node, leaf, head, inode, const_iterator, iterator, _Alloc >::operator*().

operator"!=()

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
bool __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator,
_Alloc >::operator!= (
            const _Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > & other)
const  [inline]
```

Compares content (negatively) to a different iterator object.

operator*()

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
const_reference __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _↩
CIterator, Iterator, _Alloc >::operator* () const  [inline]
```

Const access; returns the __const iterator* associated with the current leaf.

operator==()

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
bool __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator,
_Alloc >::operator== (
            const _Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > & other)
const  [inline]
```

Compares content to a different iterator object.

valid_prefix()

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
std::pair< a_const_iterator, a_const_iterator > __gnu_pbds::detail::pat_trie_base::_Node_citer<
Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::valid_prefix () const  [inline]
```

Subtree valid prefix.

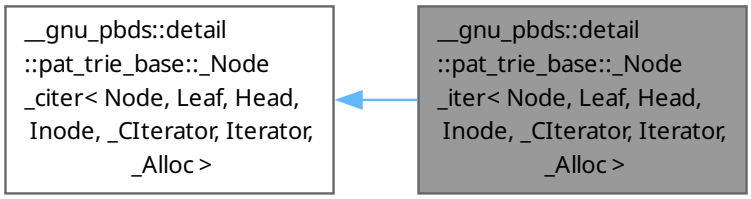
The documentation for this class was generated from the following file:

- pat_trie_base.hpp

5.131 __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > Class Template Reference

```
#include <pat_trie_base.hpp>
```

Inheritance diagram for __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >:



Public Types

- typedef value_type **const_reference**
- typedef trivial_iterator_difference_type **difference_type**
- typedef trivial_iterator_tag **iterator_category**
- typedef rebind_traits< _Alloc, metadata_type >::const_reference metadata_const_reference
- typedef Node::metadata_type metadata_type
- typedef value_type **reference**
- typedef base_type::size_type **size_type**
- typedef Iterator **value_type**

Public Member Functions

- **_Node_iter** (node_pointer p_nd=0, a_const_pointer p_traits=0)
- _Node_iter get_child (size_type i) const
- metadata_const_reference get_metadata () const
- size_type num_children () const
- bool operator!= (const _Node_citer &other) const
- reference operator* () const
- bool operator== (const _Node_citer &other) const
- std::pair< a_const_iterator, a_const_iterator > valid_prefix () const

Public Attributes

- node_pointer **m_p_nd**
- a_const_pointer **m_p_traits**

Protected Types

- typedef Node::a_const_iterator **a_const_iterator**
- typedef rebind_traits< _Alloc, Inode >::const_pointer **inode_const_pointer**
- typedef rebind_traits< _Alloc, Leaf >::const_pointer **leaf_const_pointer**
- typedef rebind_traits< _Alloc, Leaf >::pointer **leaf_pointer**

5.131.1 Detailed Description

template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename Iterator, typename _Alloc>
class __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >

Node iterator.

5.131.2 Member Typedef Documentation

metadata_const_reference

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
typedef rebind_traits<_Alloc,metadata_type>::const_reference __gnu_pbds::detail::pat_trie_base::_Node_citer<
Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::metadata_const_reference  [inherited]
```

Const metadata reference type.

metadata_type

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
typedef Node::metadata_type __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head,
Inode, _CIterator, Iterator, _Alloc >::metadata_type  [inherited]
```

Metadata type.

5.131.3 Member Function Documentation

get_child()

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
_Node_iter __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator,
```

```
Iterator, _Alloc >::get_child (
            size_type i) const  [inline]
```
Returns a node __iterator to the corresponding node's i-th child.

get_metadata()

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
metadata_const_reference __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode,
_CIterator, Iterator, _Alloc >::get_metadata () const  [inline], [inherited]
```
Metadata access.

num_children()

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
size_type __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator,
Iterator, _Alloc >::num_children () const  [inline], [inherited]
```
Returns the number of children in the corresponding node.
Referenced by __gnu_pbds::detail::pat_trie_base::_Node_citer< node, leaf, head, inode, const_iterator, iterator, _Alloc >::operator*().

operator"!=()

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
bool __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator,
_Alloc >::operator!= (
            const _Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > & other)
const  [inline], [inherited]
```
Compares content (negatively) to a different iterator object.

operator*()

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
reference __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator,
Iterator, _Alloc >::operator* () const  [inline]
```
Access; returns the iterator* associated with the current leaf.

operator==()

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
bool __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator,
_Alloc >::operator== (
            const _Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > & other)
const  [inline], [inherited]
```
Compares content to a different iterator object.

valid_prefix()

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
std::pair< a_const_iterator, a_const_iterator > __gnu_pbds::detail::pat_trie_base::_Node_citer<
Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::valid_prefix () const  [inline], [inherited]
```

Subtree valid prefix.
The documentation for this class was generated from the following file:

- pat_trie_base.hpp

5.132 __gnu_debug::_Not_equal_to< _Type > Class Template Reference

`#include <safe_sequence.h>`

Public Member Functions

- **_Not_equal_to** (const _Type &__v)
- bool **operator()** (const _Type &__x) const

5.132.1 Detailed Description

template<typename _Type>
class __gnu_debug::_Not_equal_to< _Type >

A simple function object that returns true if the passed-in value is not equal to the stored value. It saves typing over using both bind1st and not_equal.
The documentation for this class was generated from the following file:

- safe_sequence.h

5.133 std::_Not_fn< _Fn > Class Template Reference

`#include <functional>`

Public Member Functions

- template<typename _Fn2>
  constexpr **_Not_fn** (_Fn2 &&__fn, int)
- **_Not_fn** (_Not_fn &&__fn)=default
- **_Not_fn** (const _Not_fn &__fn)=default
- template<typename... _Args, typename = enable_if_t<!__is_invocable<_Fn &&, _Args...>::value>>
  void **operator()** (_Args &&... __args) &&=delete
- template<typename... _Args, typename = enable_if_t<__is_invocable<_Fn &&, _Args...>::value>>
  constexpr decltype(_S_not< __inv_res_t< _Fn &&, _Args... > >()) **operator()** (_Args &&... __args) &&noexcept(__is_nothrow_invocable< _Fn &&, _Args... >::value &&noexcept(_S_not< __inv_res_t< _Fn &&, _Args... > >()))
- template<typename... _Args, typename = enable_if_t<!__is_invocable<_Fn &, _Args...>::value>>
  void **operator()** (_Args &&... __args) &=delete
- template<typename... _Args, typename = enable_if_t<__is_invocable<_Fn &, _Args...>::value>>
  constexpr decltype(_S_not< __inv_res_t< _Fn &, _Args... > >()) **operator()** (_Args &&... __args) &noexcept(↩ __is_nothrow_invocable< _Fn &, _Args... >::value &&noexcept(_S_not< __inv_res_t< _Fn &, _Args... > >()))
- template<typename... _Args, typename = enable_if_t<!__is_invocable<_Fn const &&, _Args...>::value>>
  void **operator()** (_Args &&... __args) const &&=delete
- template<typename... _Args, typename = enable_if_t<__is_invocable<_Fn const &&, _Args...>::value>>
  constexpr decltype(_S_not< __inv_res_t< _Fn const &&, _Args... > >()) **operator()** (_Args &&... __args) const &&noexcept(__is_nothrow_invocable< _Fn const &&, _Args... >::value &&noexcept(_S_not< __inv_res_t< _Fn const &&, _Args... > >()))
- template<typename... _Args, typename = enable_if_t<!__is_invocable<_Fn const &, _Args...>::value>>
  void **operator()** (_Args &&... __args) const &=delete

- template<typename... _Args, typename = enable_if_t<__is_invocable<_Fn const &, _Args...>::value>>
  constexpr decltype(_S_not< __inv_res_t< _Fn const &, _Args... > >()) **operator()** (_Args &&... __args) const &noexcept(__is_nothrow_invocable< _Fn const &, _Args... >::value &&noexcept(_S_not< __inv_res_t< _Fn const &, _Args... > >()))

5.133.1 Detailed Description

template<typename _Fn>
class std::_Not_fn< _Fn >

Generalized negator.
The documentation for this class was generated from the following file:

- functional

5.134 __gnu_parallel::_Nothing Struct Reference

`#include <for_each_selectors.h>`

Public Member Functions

- template<typename _It>
  void operator() (_It __i)

5.134.1 Detailed Description

Functor doing nothing.
For some __reduction tasks (this is not a function object, but is passed as __selector __dummy parameter.

5.134.2 Member Function Documentation

operator()()

```
template<typename _It>
void __gnu_parallel::_Nothing::operator() (
             _It __i)  [inline]
```

Functor execution.

Parameters

↩ _↩ ↩ _↩ *i*	iterator referencing object.

The documentation for this struct was generated from the following file:

- for_each_selectors.h

5.135 __gnu_parallel::_Piece< _DifferenceTp > Struct Template Reference

`#include <multiway_mergesort.h>`

Public Types

- typedef _DifferenceTp **_DifferenceType**

Public Attributes

- _DifferenceType _M_begin
- _DifferenceType _M_end

5.135.1 Detailed Description

template<typename _DifferenceTp>
struct __gnu_parallel::_Piece< _DifferenceTp >

Subsequence description.

5.135.2 Member Data Documentation

_M_begin

```
template<typename _DifferenceTp>
_DifferenceType __gnu_parallel::_Piece< _DifferenceTp >::_M_begin
```

Begin of subsequence.

_M_end

```
template<typename _DifferenceTp>
_DifferenceType __gnu_parallel::_Piece< _DifferenceTp >::_M_end
```

End of subsequence.
The documentation for this struct was generated from the following file:

- multiway_mergesort.h

5.136 std::_Placeholder< _Num > Struct Template Reference

```
#include <functional>
```

5.136.1 Detailed Description

template<int _Num>
struct std::_Placeholder< _Num >

The type of placeholder objects defined by libstdc++.

Since

C++11

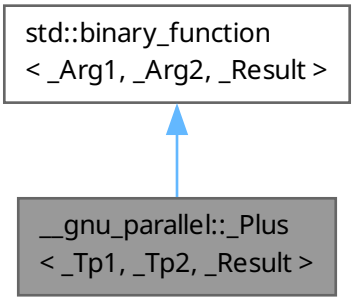
The documentation for this struct was generated from the following file:

- functional

5.137 __gnu_parallel::_Plus< _Tp1, _Tp2, _Result > Struct Template Reference

```
#include <base.h>
```

Inheritance diagram for __gnu_parallel::_Plus< _Tp1, _Tp2, _Result >:



Public Types

- typedef _Arg1 first_argument_type
- typedef _Result result_type
- typedef _Arg2 second_argument_type

Public Member Functions

- _Result **operator()** (const _Tp1 &__x, const _Tp2 &__y) const

5.137.1 Detailed Description

template<typename _Tp1, typename _Tp2, typename _Result = __typeof__(*static_cast<_Tp1*>(0) + *static_cast<_Tp2*>(0))>
struct __gnu_parallel::_Plus< _Tp1, _Tp2, _Result >

Similar to std::plus, but allows two different types.

5.137.2 Member Typedef Documentation

first_argument_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result >::first_argument_type  [inherited]
```

`first_argument_type` is the type of the first argument

result_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Result std::binary_function< _Arg1, _Arg2, _Result >::result_type  [inherited]
```

`result_type` is the return type

second_argument_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result >::second_argument_type  [inherited]
```

second_argument_type is the type of the second argument
The documentation for this struct was generated from the following file:

- base.h

5.138 __gnu_parallel::_PMWMSSortingData< _RAIter > Struct Template Reference

#include <multiway_mergesort.h>

Public Types

- typedef _TraitsType::difference_type **_DifferenceType**
- typedef std::iterator_traits< _RAIter > **_TraitsType**
- typedef _TraitsType::value_type **_ValueType**

Public Attributes

- _ThreadIndex _M_num_threads
- _DifferenceType * _M_offsets
- std::vector< _Piece< _DifferenceType > > * _M_pieces
- _ValueType * _M_samples
- _RAIter _M_source
- _DifferenceType * _M_starts
- _ValueType ** _M_temporary

5.138.1 Detailed Description

template<typename _RAIter>
struct __gnu_parallel::_PMWMSSortingData< _RAIter >

Data accessed by all threads.
PMWMS = parallel multiway mergesort

5.138.2 Member Data Documentation

_M_num_threads

```
template<typename _RAIter>
_ThreadIndex __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_num_threads
```

Number of threads involved.
Referenced by __gnu_parallel::parallel_sort_mwms(), and __gnu_parallel::parallel_sort_mwms_pu().

_M_offsets

```
template<typename _RAIter>
_DifferenceType* __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_offsets
```

Offsets to add to the found positions.
Referenced by __gnu_parallel::parallel_sort_mwms().

_M_pieces

```
template<typename _RAIter>
std::vector<_Piece<_DifferenceType> >* __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_pieces
```

Pieces of data to merge [thread][__sequence].
Referenced by __gnu_parallel::parallel_sort_mwms(), and __gnu_parallel::parallel_sort_mwms_pu().

_M_samples

```
template<typename _RAIter>
_ValueType* __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_samples
```

Samples.
Referenced by __gnu_parallel::__determine_samples(), and __gnu_parallel::parallel_sort_mwms().

_M_source

```
template<typename _RAIter>
_RAIter __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_source
```

Input __begin.
Referenced by __gnu_parallel::__determine_samples(), __gnu_parallel::parallel_sort_mwms(), and __gnu_parallel::parallel_sort_mwms_

_M_starts

```
template<typename _RAIter>
_DifferenceType* __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_starts
```

Start indices, per thread.
Referenced by __gnu_parallel::__determine_samples(), __gnu_parallel::parallel_sort_mwms(), and __gnu_parallel::parallel_sort_mwms_

_M_temporary

```
template<typename _RAIter>
_ValueType** __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_temporary
```

Storage in which to sort.
Referenced by __gnu_parallel::parallel_sort_mwms(), and __gnu_parallel::parallel_sort_mwms_pu().
The documentation for this struct was generated from the following file:

- multiway_mergesort.h

5.139 __gnu_cxx::_Pointer_adapter< _Storage_policy > Class Template Reference

```
#include <pointer.h>
```

Public Types

- typedef std::ptrdiff_t **difference_type**
- typedef _Storage_policy::element_type **element_type**
- typedef std::random_access_iterator_tag **iterator_category**
- typedef _Pointer_adapter **pointer**
- typedef _Reference_type< element_type >::reference **reference**
- typedef _Unqualified_type< element_type >::type **value_type**

Public Member Functions

- template<typename _Up>
  _Pointer_adapter (_Up *__arg)
- **_Pointer_adapter** (const _Pointer_adapter &__arg)
- template<typename _Up>
  _Pointer_adapter (const _Pointer_adapter< _Up > &__arg)
- **_Pointer_adapter** (element_type *__arg=0)
- **operator bool** () const
- reference **operator*** () const
- _Pointer_adapter & **operator++** ()

- _Pointer_adapter **operator++** (int)
- _Pointer_adapter & **operator+=** (int __offset)
- _Pointer_adapter & **operator+=** (long __offset)
- _Pointer_adapter & **operator+=** (long long __offset)
- _Pointer_adapter & **operator+=** (short __offset)
- _Pointer_adapter & **operator+=** (unsigned int __offset)
- _Pointer_adapter & **operator+=** (unsigned long __offset)
- _Pointer_adapter & **operator+=** (unsigned long long __offset)
- _Pointer_adapter & **operator+=** (unsigned short __offset)
- template<typename _Up>
  std::ptrdiff_t **operator-** (const _Pointer_adapter< _Up > &__rhs) const
- _Pointer_adapter & **operator--** ()
- _Pointer_adapter **operator--** (int)
- _Pointer_adapter & **operator-=** (int __offset)
- _Pointer_adapter & **operator-=** (long __offset)
- _Pointer_adapter & **operator-=** (long long __offset)
- _Pointer_adapter & **operator-=** (short __offset)
- _Pointer_adapter & **operator-=** (unsigned int __offset)
- _Pointer_adapter & **operator-=** (unsigned long __offset)
- _Pointer_adapter & **operator-=** (unsigned long long __offset)
- _Pointer_adapter & **operator-=** (unsigned short __offset)
- element_type * **operator->** () const
- template<typename _Up>
  _Pointer_adapter & **operator=** (_Up *__arg)
- _Pointer_adapter & **operator=** (const _Pointer_adapter &__arg)
- template<typename _Up>
  _Pointer_adapter & **operator=** (const _Pointer_adapter< _Up > &__arg)
- reference **operator[ ]** (std::ptrdiff_t __index) const

Friends

- _Pointer_adapter **operator+** (const _Pointer_adapter &__lhs, int __offset)
- _Pointer_adapter **operator+** (const _Pointer_adapter &__lhs, long __offset)
- _Pointer_adapter **operator+** (const _Pointer_adapter &__lhs, long long __offset)
- _Pointer_adapter **operator+** (const _Pointer_adapter &__lhs, short __offset)
- _Pointer_adapter **operator+** (const _Pointer_adapter &__lhs, unsigned int __offset)
- _Pointer_adapter **operator+** (const _Pointer_adapter &__lhs, unsigned long __offset)
- _Pointer_adapter **operator+** (const _Pointer_adapter &__lhs, unsigned long long __offset)
- _Pointer_adapter **operator+** (const _Pointer_adapter &__lhs, unsigned short __offset)
- _Pointer_adapter **operator+** (int __offset, const _Pointer_adapter &__rhs)
- _Pointer_adapter **operator+** (long __offset, const _Pointer_adapter &__rhs)
- _Pointer_adapter **operator+** (long long __offset, const _Pointer_adapter &__rhs)
- _Pointer_adapter **operator+** (short __offset, const _Pointer_adapter &__rhs)
- _Pointer_adapter **operator+** (unsigned int __offset, const _Pointer_adapter &__rhs)
- _Pointer_adapter **operator+** (unsigned long __offset, const _Pointer_adapter &__rhs)
- _Pointer_adapter **operator+** (unsigned long long __offset, const _Pointer_adapter &__rhs)
- _Pointer_adapter **operator+** (unsigned short __offset, const _Pointer_adapter &__rhs)
- template<typename _Up>
  std::ptrdiff_t **operator-** (_Up *__lhs, const _Pointer_adapter &__rhs)
- template<typename _Up>
  std::ptrdiff_t **operator-** (const _Pointer_adapter &__lhs, _Up *__rhs)

- std::ptrdiff_t **operator-** (const _Pointer_adapter &__lhs, element_type *__rhs)
- _Pointer_adapter **operator-** (const _Pointer_adapter &__lhs, int __offset)
- _Pointer_adapter **operator-** (const _Pointer_adapter &__lhs, long __offset)
- _Pointer_adapter **operator-** (const _Pointer_adapter &__lhs, long long __offset)
- _Pointer_adapter **operator-** (const _Pointer_adapter &__lhs, short __offset)
- _Pointer_adapter **operator-** (const _Pointer_adapter &__lhs, unsigned int __offset)
- _Pointer_adapter **operator-** (const _Pointer_adapter &__lhs, unsigned long __offset)
- _Pointer_adapter **operator-** (const _Pointer_adapter &__lhs, unsigned long long __offset)
- _Pointer_adapter **operator-** (const _Pointer_adapter &__lhs, unsigned short __offset)
- std::ptrdiff_t **operator-** (element_type *__lhs, const _Pointer_adapter &__rhs)
- std::strong_ordering **operator<=>** (const _Pointer_adapter &__lhs, const _Pointer_adapter &__rhs) noexcept

5.139.1 Detailed Description

template<typename _Storage_policy>
class __gnu_cxx::_Pointer_adapter< _Storage_policy >

The following provides an 'alternative pointer' that works with the containers when specified as the pointer typedef of the allocator.
The pointer type used with the containers doesn't have to be this class, but it must support the implicit conversions, pointer arithmetic, comparison operators, etc. that are supported by this class, and avoid raising compile-time ambiguities. Because creating a working pointer can be challenging, this pointer template was designed to wrapper an easier storage policy type, so that it becomes reusable for creating other pointer types.
A key point of this class is also that it allows container writers to 'assume' Allocator::pointer is a typedef for a normal pointer. This class supports most of the conventions of a true pointer, and can, for instance handle implicit conversion to const and base class pointer types. The only impositions on container writers to support extended pointers are: 1) use the Allocator::pointer typedef appropriately for pointer types. 2) if you need pointer casting, use the __pointer_cast<> functions from ext/cast.h. This allows pointer cast operations to be overloaded as necessary by custom pointers.
Note: The const qualifier works with this pointer adapter as follows:
_Tp* == _Pointer_adapter<_Std_pointer_impl<_Tp> >; const _Tp* == _Pointer_adapter<_Std_pointer_impl<const _Tp> >; _Tp* const == const _Pointer_adapter<_Std_pointer_impl<_Tp> >; const _Tp* const == const _Pointer_adapter<_Std_pointer_impl<const _Tp> >;
The documentation for this class was generated from the following file:

- pointer.h

5.140 __gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp > Class Template Reference

`#include <base.h>`

Public Types

- typedef _DifferenceTp **_DifferenceType**
- typedef _PseudoSequenceIterator< _Tp, uint64_t > **iterator**

Public Member Functions

- _PseudoSequence (const _Tp &__val, _DifferenceType __count)
- iterator begin () const
- iterator end () const

5.140.1 Detailed Description

template<typename _Tp, typename _DifferenceTp>
class __gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >

Sequence that conceptually consists of multiple copies of the same element. The copies are not stored explicitly, of course.

Parameters

_Tp	Sequence _M_value type.
_DifferenceTp	Sequence difference type.

5.140.2 Constructor & Destructor Documentation

_PseudoSequence()

```
template<typename _Tp, typename _DifferenceTp>
__gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >::_PseudoSequence (
            const _Tp & __val,
            _DifferenceType __count)  [inline]
```

Constructor.

Parameters

__val	Element of the sequence.
__count	Number of (virtual) copies.

5.140.3 Member Function Documentation

begin()

```
template<typename _Tp, typename _DifferenceTp>
iterator __gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >::begin () const  [inline]
```

Begin iterator.

end()

```
template<typename _Tp, typename _DifferenceTp>
iterator __gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >::end () const  [inline]
```

End iterator.

The documentation for this class was generated from the following file:

- base.h

5.141 __gnu_parallel::_PseudoSequenceIterator< _Tp, _DifferenceTp > Class Template Reference

```
#include <base.h>
```

Public Types

- typedef _DifferenceTp **_DifferenceType**

Public Member Functions

- **_PseudoSequenceIterator** (const _Tp &__val, _DifferenceType __pos)
- bool **operator!=** (const _PseudoSequenceIterator &__i2)
- const _Tp & **operator*** () const
- _PseudoSequenceIterator & **operator++** ()
- _PseudoSequenceIterator **operator++** (int)
- _DifferenceType **operator-** (const _PseudoSequenceIterator &__i2)
- bool **operator==** (const _PseudoSequenceIterator &__i2)
- const _Tp & **operator[]** (_DifferenceType) const

5.141.1 Detailed Description

template<typename _Tp, typename _DifferenceTp>
class __gnu_parallel::_PseudoSequenceIterator< _Tp, _DifferenceTp >

_Iterator associated with __gnu_parallel::_PseudoSequence. If features the usual random-access iterator functionality.

Parameters

_Tp	Sequence _M_value type.
_DifferenceTp	Sequence difference type.

The documentation for this class was generated from the following file:

- base.h

5.142 __gnu_parallel::_QSBThreadLocal< _RAIter > Struct Template Reference

`#include <balanced_quicksort.h>`

Public Types

- typedef _TraitsType::difference_type **_DifferenceType**
- typedef std::pair< _RAIter, _RAIter > _Piece
- typedef std::iterator_traits< _RAIter > **_TraitsType**

Public Member Functions

- _QSBThreadLocal (int __queue_size)

Public Attributes

- volatile _DifferenceType * _M_elements_leftover
- _Piece _M_global
- _Piece _M_initial
- _RestrictedBoundedConcurrentQueue< _Piece > _M_leftover_parts
- _ThreadIndex _M_num_threads

5.142.1 Detailed Description

template<typename _RAIter>
struct __gnu_parallel::_QSBThreadLocal< _RAIter >

Information local to one thread in the parallel quicksort run.

5.142.2 Member Typedef Documentation

_Piece

```
template<typename _RAIter>
typedef std::pair<_RAIter, _RAIter> __gnu_parallel::_QSBThreadLocal< _RAIter >::_Piece
```

Continuous part of the sequence, described by an iterator pair.

5.142.3 Constructor & Destructor Documentation

_QSBThreadLocal()

```
template<typename _RAIter>
__gnu_parallel::_QSBThreadLocal< _RAIter >::_QSBThreadLocal (
            int __queue_size)  [inline]
```

Constructor.

Parameters

__queue_size	size of the work-stealing queue.

References _M_leftover_parts.

5.142.4 Member Data Documentation

_M_elements_leftover

```
template<typename _RAIter>
volatile _DifferenceType* __gnu_parallel::_QSBThreadLocal< _RAIter >::_M_elements_leftover
```

Pointer to a counter of elements left over to sort.
Referenced by __gnu_parallel::__qsb_conquer(), and __gnu_parallel::__qsb_local_sort_with_helping().

_M_global

```
template<typename _RAIter>
_Piece __gnu_parallel::_QSBThreadLocal< _RAIter >::_M_global
```

The complete sequence to sort.

_M_initial

```
template<typename _RAIter>
_Piece __gnu_parallel::_QSBThreadLocal< _RAIter >::_M_initial
```

Initial piece to work on.
Referenced by __gnu_parallel::__qsb_conquer(), and __gnu_parallel::__qsb_local_sort_with_helping().

_M_leftover_parts

```
template<typename _RAIter>
_RestrictedBoundedConcurrentQueue<_Piece> __gnu_parallel::_QSBThreadLocal< _RAIter >::_M_↩
leftover_parts
```

Work-stealing queue.
Referenced by _QSBThreadLocal(), and __gnu_parallel::__qsb_local_sort_with_helping().

_M_num_threads

```
template<typename _RAIter>
_ThreadIndex __gnu_parallel::_QSBThreadLocal< _RAIter >::_M_num_threads
```

Number of threads involved in this algorithm.
Referenced by __gnu_parallel::__qsb_local_sort_with_helping().
The documentation for this struct was generated from the following file:

- balanced_quicksort.h

5.143 std::__detail::_Quoted_string< _String, _CharT > Struct Template Reference

```
#include <quoted_string.h>
```

Public Member Functions

- **_Quoted_string** (_String __str, _CharT __del, _CharT __esc)
- _Quoted_string & **operator=** (_Quoted_string &)=delete

Public Attributes

- _CharT **_M_delim**
- _CharT **_M_escape**
- _String **_M_string**

5.143.1 Detailed Description

template<typename _String, typename _CharT>
struct std::__detail::_Quoted_string< _String, _CharT >

Struct for delimited strings.
The documentation for this struct was generated from the following file:

- quoted_string.h

5.144 __gnu_parallel::_RandomNumber Class Reference

```
#include <random_number.h>
```

Public Member Functions

- _RandomNumber ()
- _RandomNumber (uint32_t __seed, uint64_t _M_supremum=0x100000000ULL)
- unsigned long __genrand_bits (int __bits)
- uint32_t operator() ()
- uint32_t operator() (uint64_t local_supremum)

5.144.1 Detailed Description

Random number generator, based on the Mersenne twister.

5.144.2 Constructor & Destructor Documentation

_RandomNumber() [1/2]

```
__gnu_parallel::_RandomNumber::_RandomNumber ()  [inline]
```

Default constructor. Seed with 0.

_RandomNumber() [2/2]

```
__gnu_parallel::_RandomNumber::_RandomNumber (
            uint32_t __seed,
            uint64_t _M_supremum = 0x100000000ULL)  [inline]
```

Constructor.

Parameters

__seed	Random __seed.
_M_supremum	Generate integer random numbers in the interval [0,_M_supremum).

5.144.3 Member Function Documentation

__genrand_bits()

```
unsigned long __gnu_parallel::_RandomNumber::__genrand_bits (
            int __bits)  [inline]
```

Generate a number of random bits, run-time parameter.

Parameters

__bits	Number of bits to generate.

operator()() [1/2]

```
uint32_t __gnu_parallel::_RandomNumber::operator() ()  [inline]
```

Generate unsigned random 32-bit integer.

operator()() [2/2]

```
uint32_t __gnu_parallel::_RandomNumber::operator() (
            uint64_t local_supremum)  [inline]
```

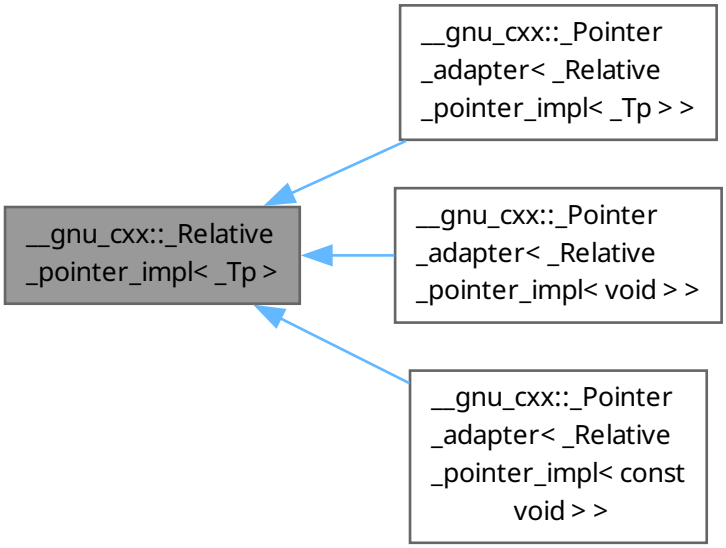
Generate unsigned random 32-bit integer in the interval [0,local_supremum).
The documentation for this class was generated from the following file:

- random_number.h

5.145 __gnu_cxx::_Relative_pointer_impl< _Tp > Class Template Reference

```
#include <pointer.h>
```

Inheritance diagram for __gnu_cxx::_Relative_pointer_impl< _Tp >:



Public Types

- typedef _Tp **element_type**

Public Member Functions

- _Tp * **get** () const
- bool **operator<** (const _Relative_pointer_impl &__rarg) const
- bool **operator==** (const _Relative_pointer_impl &__rarg) const
- void **set** (_Tp *__arg)

5.145.1 Detailed Description

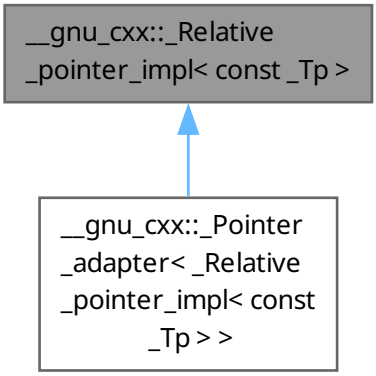
template<typename _Tp>
class __gnu_cxx::_Relative_pointer_impl< _Tp >

A storage policy for use with _Pointer_adapter<> which stores the pointer's address as an offset value which is relative to its own address.
This is intended for pointers within shared memory regions which might be mapped at different addresses by different processes. For null pointers, a value of 1 is used. (0 is legitimate sometimes for nodes in circularly linked lists) This value was chosen as the least likely to generate an incorrect null, As there is no reason why any normal pointer would point 1 byte into its own pointer address.
The documentation for this class was generated from the following file:

- pointer.h

5.146 __gnu_cxx::_Relative_pointer_impl< const _Tp > Class Template Reference

`#include <pointer.h>`
Inheritance diagram for __gnu_cxx::_Relative_pointer_impl< const _Tp >:



Public Types

- typedef _Tp **element_type**
- typedef const _Tp **element_type**

Public Member Functions

- _Tp * **get** () const
- const _Tp * **get** () const
- bool **operator<** (const _Relative_pointer_impl &__rarg) const
- bool **operator<** (const _Relative_pointer_impl &__rarg) const
- bool **operator==** (const _Relative_pointer_impl &__rarg) const
- bool **operator==** (const _Relative_pointer_impl &__rarg) const
- void **set** (_Tp *__arg)
- void **set** (const _Tp *__arg)

5.146.1 Detailed Description

template<typename _Tp>
class __gnu_cxx::_Relative_pointer_impl< const _Tp >

Relative_pointer_impl needs a specialization for const T because of the casting done during pointer arithmetic.
The documentation for this class was generated from the following file:

- pointer.h

5.147 __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp > Class Template Reference

`#include <queue.h>`

Public Member Functions

- _RestrictedBoundedConcurrentQueue (_SequenceIndex __max_size)
- ~_RestrictedBoundedConcurrentQueue ()
- bool pop_back (_Tp &__t)
- bool pop_front (_Tp &__t)
- void push_front (const _Tp &__t)

5.147.1 Detailed Description

template<typename _Tp>
class __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >

Double-ended queue of bounded size, allowing lock-free atomic access. push_front() and pop_front() must not be called concurrently to each other, while pop_back() can be called concurrently at all times. `empty()`, `size()`, and `top()` are intentionally not provided. Calling them would not make sense in a concurrent setting.

Parameters

_Tp	Contained element type.

5.147.2 Constructor & Destructor Documentation

_RestrictedBoundedConcurrentQueue()

```
template<typename _Tp>
__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::_RestrictedBoundedConcurrentQueue (
            _SequenceIndex __max_size)  [inline]
```

Constructor. Not to be called concurrent, of course.

Parameters

__max_size	Maximal number of elements to be contained.

References __gnu_parallel::__encode2().

~_RestrictedBoundedConcurrentQueue()

```
template<typename _Tp>
__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::~_RestrictedBoundedConcurrentQueue ()
[inline]
```

Destructor. Not to be called concurrent, of course.

5.147.3 Member Function Documentation

pop_back()

```
template<typename _Tp>
bool __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::pop_back (
            _Tp & __t)  [inline]
```

Pops one element from the queue at the front end. Must not be called concurrently with pop_front().
References __gnu_parallel::__compare_and_swap(), __gnu_parallel::__decode2(), and __gnu_parallel::__encode2().

pop_front()

```
template<typename _Tp>
bool __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::pop_front (
            _Tp & __t)  [inline]
```

Pops one element from the queue at the front end. Must not be called concurrently with pop_front().
References __gnu_parallel::__compare_and_swap(), __gnu_parallel::__decode2(), and __gnu_parallel::__encode2().

push_front()

```
template<typename _Tp>
void __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::push_front (
            const _Tp & __t)  [inline]
```

Pushes one element into the queue at the front end. Must not be called concurrently with pop_front().
References __gnu_parallel::__decode2(), __gnu_parallel::__encode2(), and __gnu_parallel::__fetch_and_add().
The documentation for this class was generated from the following file:

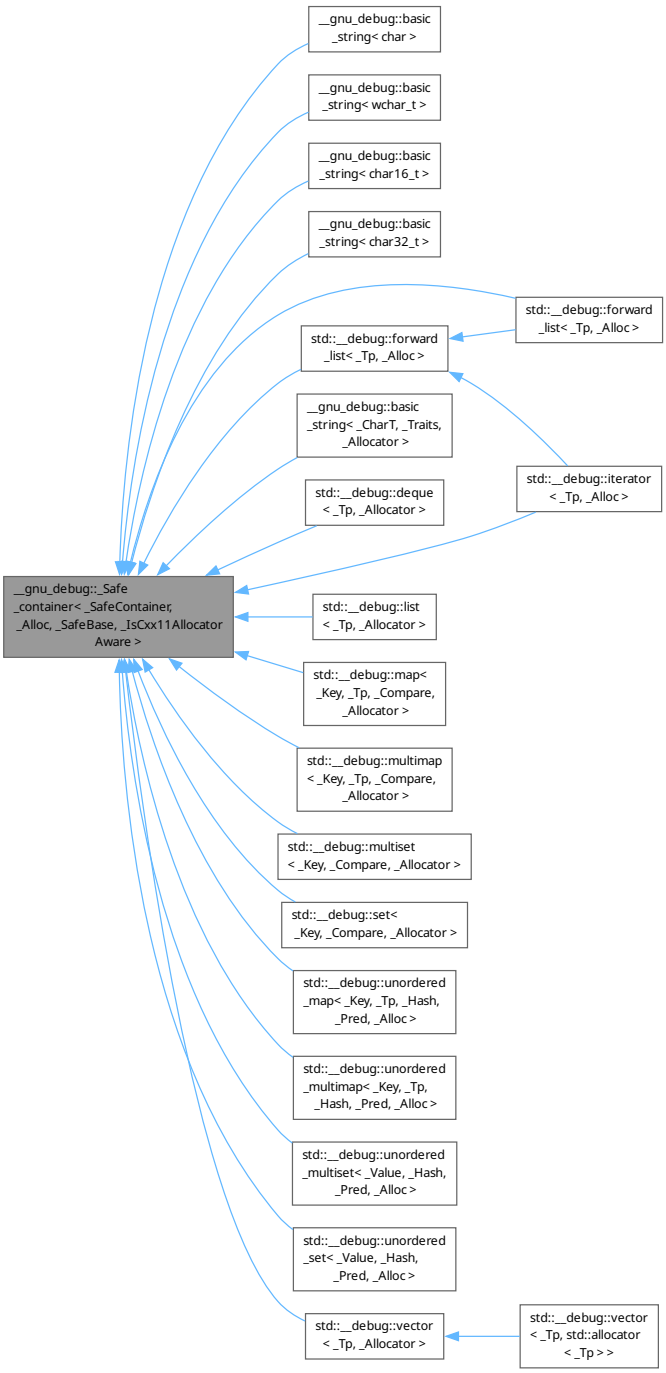
- queue.h

5.148 __gnu_debug::_Safe_container< _SafeContainer, _Alloc, _SafeBase, _IsCxx11AllocatorAware > Class Template Reference

```
#include <safe_container.h>
```

Inheritance diagram for __gnu_debug::_Safe_container< _SafeContainer, _Alloc, _SafeBase, _IsCxx11AllocatorAware

>:



Protected Member Functions

- **_Safe_container** (_Safe_container &&)=default
- constexpr **_Safe_container** (_Safe_container &&__x, const _Alloc &__a)
- **_Safe_container** (const _Safe_container &)=default

- constexpr void **_M_swap** (const _Safe_container &__x) const noexcept
- constexpr _Safe_container & **operator=** (_Safe_container &&__x) noexcept
- constexpr _Safe_container & **operator=** (const _Safe_container &) noexcept

5.148.1 Detailed Description

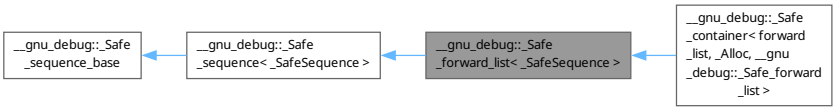
template<typename _SafeContainer, typename _Alloc, template< typename > class _SafeBase, bool _IsCxx11AllocatorAware = true>
class __gnu_debug::_Safe_container< _SafeContainer, _Alloc, _SafeBase, _IsCxx11AllocatorAware >

Safe class dealing with some allocator dependent operations.
The documentation for this class was generated from the following file:

- safe_container.h

5.149 __gnu_debug::_Safe_forward_list< _SafeSequence > Class Template Reference

`#include <forward_list>`
Inheritance diagram for __gnu_debug::_Safe_forward_list< _SafeSequence >:



Public Member Functions

- void _M_invalidate_if (_Predicate __pred) const
- void _M_transfer_from_if (const _Safe_sequence &__from, _Predicate __pred) const

Public Attributes

- _Safe_iterator_base * _M_const_iterators
- _Safe_iterator_base * _M_iterators
- unsigned int _M_version

Protected Member Functions

- void _M_detach_all () const
- void _M_detach_singular () const
- __gnu_cxx::__mutex & _M_get_mutex () const noexcept
- void **_M_invalidate_all** () const
- void _M_revalidate_singular () const
- void **_M_swap** (const _Safe_forward_list &) const noexcept
- void _M_swap (const _Safe_sequence_base &__x) const noexcept

5.149.1 Detailed Description

template<typename _SafeSequence>
class __gnu_debug::_Safe_forward_list< _SafeSequence >

Special iterators swap and invalidation for forward_list because of the before_begin iterator.

5.149.2 Member Function Documentation

_M_detach_all()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_all () const  [protected], [inherited]
```
Detach all iterators, leaving them singular.

_M_detach_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular () const  [protected], [inherited]
```
Detach all singular iterators.

Postcondition

for all iterators i attached to this sequence, i->_M_version == _M_version.

_M_get_mutex()

```
__gnu_cxx::__mutex & __gnu_debug::_Safe_sequence_base::_M_get_mutex () const  [protected], [noexcept],
[inherited]
```
For use in _Safe_sequence.

_M_invalidate_if()

```
void __gnu_debug::_Safe_sequence< _SafeSequence >::_M_invalidate_if (
            _Predicate __pred) const  [inherited]
```
Invalidates all iterators `x` that reference this sequence, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal iterators nested in the safe ones.
References __gnu_debug::_Safe_iterator_base::_M_next, and __gnu_debug::_Safe_iterator_base::_M_prior.

_M_revalidate_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () const  [protected], [inherited]
```
Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

_M_swap()

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
            const _Safe_sequence_base & __x) const  [protected], [noexcept], [inherited]
```
Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

_M_transfer_from_if()

```
void __gnu_debug::_Safe_sequence< _SafeSequence >::_M_transfer_from_if (
            const _Safe_sequence< _SafeSequence > & __from,
            _Predicate __pred) const  [inherited]
```
Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal iterators nested in the safe ones.

5.149.3 Member Data Documentation

_M_const_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators  [mutable], [inherited]
```
The list of constant iterators that reference this container.

_M_iterators

_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [mutable], [inherited]
The list of mutable iterators that reference this container.

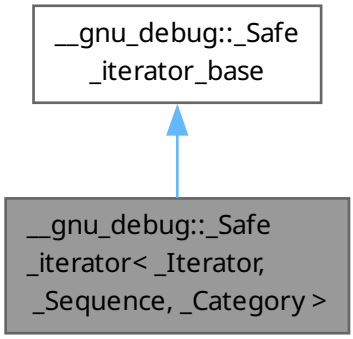
_M_version

unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable], [inherited]
The container version number. This number may never be 0.
The documentation for this class was generated from the following file:

- debug/forward_list

5.150 __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category > Class Template Reference

#include <safe_iterator.h>
Inheritance diagram for __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >:



Public Types

- typedef _Safe_iterator< _Iterator, _Sequence, iterator_category > **_Self**
- typedef _Traits::difference_type **difference_type**
- typedef _Traits::iterator_category **iterator_category**
- typedef _Iterator **iterator_type**
- typedef _Traits::pointer **pointer**
- typedef _Traits::reference **reference**
- typedef _Traits::value_type **value_type**

Public Member Functions

- constexpr _Safe_iterator () noexcept
- constexpr _Safe_iterator (_Iterator __i, const _Safe_sequence_base *__seq) noexcept
- constexpr _Safe_iterator (_Safe_iterator &&__x) noexcept
- constexpr _Safe_iterator (const _Safe_iterator &__x) noexcept

- template<typename _MutableIterator>
  constexpr _Safe_iterator (const _Safe_iterator< _MutableIterator, _Sequence, typename __gnu_cxx::__enable↩
  _if< _IsConstant::__value &&std::__are_same< _MutableIterator, _OtherIterator >::__value, _Category >::_↩
  _type > &__x) noexcept
- void _M_attach (const _Safe_sequence_base *__seq)
- void _M_attach_single (const _Safe_sequence_base *__seq)
- bool _M_attached_to (const _Safe_sequence_base *__seq) const
- bool _M_before_dereferenceable () const
- template<typename _Diff>
  bool **_M_can_advance** (const std::pair< _Diff, _Distance_precision > &__dist, int __way) const
- bool **_M_can_advance** (difference_type __n, bool __strict=false) const
- bool _M_can_compare (const _Safe_iterator_base &__x) const noexcept
- bool _M_dereferenceable () const
- void _M_detach_single () noexcept
- _Distance_traits< _Iterator >::__type **_M_get_distance_from_begin** () const
- _Distance_traits< _Iterator >::__type **_M_get_distance_to** (const _Safe_iterator &__rhs) const
- _Distance_traits< _Iterator >::__type **_M_get_distance_to_end** () const
- __gnu_cxx::__conditional_type< _IsConstant::__value, const_Sequence *, _Sequence * >::__type **_M_get_↩
  sequence** () const
- bool _M_incrementable () const
- void _M_invalidate ()
- bool _M_is_before_begin () const
- constexpr bool _M_is_begin () const
- bool _M_is_beginnest () const
- bool _M_is_end () const
- void _M_reset () noexcept
- bool _M_singular () const noexcept
- void _M_unlink () noexcept
- bool **_M_valid_range** (const _Safe_iterator &__rhs, std::pair< difference_type, _Distance_precision > &__dist, bool __check_dereferenceable=true) const
- bool _M_value_initialized () const
- constexpr const _Iterator & **base** () const noexcept
- constexpr _Iterator & base () noexcept
- constexpr operator _Iterator () const noexcept
- constexpr reference operator* () const noexcept
- constexpr _Safe_iterator & operator++ () noexcept
- constexpr _Safe_iterator operator++ (int) noexcept
- constexpr pointer operator-> () const noexcept
- constexpr _Safe_iterator & operator= (_Safe_iterator &&__x) noexcept
- constexpr _Safe_iterator & operator= (const _Safe_iterator &__x) noexcept

Static Public Member Functions

- static constexpr bool _S_constant ()

Public Attributes

- _Safe_iterator_base * _M_next
- _Safe_iterator_base * _M_prior
- const _Safe_sequence_base * _M_sequence
- unsigned int _M_version

Protected Types

- typedef std::__are_same< typename _Sequence::_Base::const_iterator, _Iterator > **_IsConstant**
- typedef __gnu_cxx::__conditional_type< _IsConstant::__value, typename _Sequence::_Base::iterator, typename↩ _Sequence::_Base::const_iterator >::__type **_OtherIterator**

Protected Member Functions

- constexpr **_Safe_iterator** (const _Safe_iterator &__x, _Unchecked) noexcept
- void _M_attach (const _Safe_sequence_base *__seq, bool __constant)
- void _M_attach_single (const _Safe_sequence_base *__seq, bool __constant) noexcept
- void _M_detach ()
- __gnu_cxx::__mutex & _M_get_mutex () noexcept

Friends

- template<typename _IteR>
  constexpr friend bool **operator==** (const _Self &__lhs, const _Safe_iterator< _IteR, _Sequence, iterator_category > &__rhs) noexcept
- constexpr friend bool **operator==** (const _Self &__lhs, const _Self &__rhs) noexcept

5.150.1 Detailed Description

template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_↩ Iterator>::iterator_category>
class __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >

Safe iterator wrapper.
The class template _Safe_iterator is a wrapper around an iterator that tracks the iterator's movement among sequences and checks that operations performed on the "safe" iterator are legal. In additional to the basic iterator operations (which are validated, and then passed to the underlying iterator), _Safe_iterator has member functions for iterator invalidation, attaching/detaching the iterator from sequences, and querying the iterator's state.
Note that _Iterator must be the first base class so that it gets initialized before the iterator is being attached to the container's list of iterators and it is being detached before _Iterator get destroyed. Otherwise it would result in a data race.

5.150.2 Constructor & Destructor Documentation

_Safe_iterator() [1/5]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_Safe_iterator ()  [inline],
[constexpr], [noexcept]
```

Postcondition

the iterator is singular and unattached

_Safe_iterator() [2/5]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_Safe_iterator (
            _Iterator __i,
            const _Safe_sequence_base * __seq)  [inline], [constexpr], [noexcept]
```

Safe iterator construction from an unsafe iterator and its sequence.

Precondition

seq is not NULL

Postcondition

this is not singular

_Safe_iterator() [3/5]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_Safe_iterator (
            const _Safe_iterator< _Iterator, _Sequence, _Category > & __x)  [inline], [constexpr],
[noexcept]
```

Copy construction.

_Safe_iterator() [4/5]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_Safe_iterator (
            _Safe_iterator< _Iterator, _Sequence, _Category > && __x)  [inline], [constexpr],
[noexcept]
```

Move construction.

Postcondition

__x is singular and unattached

_Safe_iterator() [5/5]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
template<typename _MutableIterator>
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_Safe_iterator (
            const _Safe_iterator< _MutableIterator, _Sequence, typename __gnu_cxx::__enable_if<
_IsConstant::__value &&std::__are_same< _MutableIterator, _OtherIterator >::__value, _Category
>::__type > & __x)  [inline], [constexpr], [noexcept]
```

Converting constructor from a mutable iterator to a constant iterator.

5.150.3 Member Function Documentation

_M_attach() [1/2]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
void __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_M_attach (
            const _Safe_sequence_base * __seq)  [inline]
```

Attach iterator to the given sequence.

Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), __gnu_debug::_Safe_iterator< _Base_iterator, m
__gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator
and __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=().

_M_attach() [2/2]

```
void __gnu_debug::_Safe_iterator_base::_M_attach (
            const _Safe_sequence_base * __seq,
            bool __constant)  [protected], [inherited]
```

Attaches this iterator to the given sequence, detaching it from whatever sequence it was attached to originally. If the new sequence is the NULL pointer, the iterator is left unattached.
References _M_attach().
Referenced by _Safe_iterator_base(), _Safe_iterator_base(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_attach(), and _M_attach().

_M_attach_single() [1/2]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
void __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_M_attach_single (
            const _Safe_sequence_base * __seq)  [inline]
```

Likewise, but not thread-safe.

_M_attach_single() [2/2]

```
void __gnu_debug::_Safe_iterator_base::_M_attach_single (
            const _Safe_sequence_base * __seq,
            bool __constant)  [protected], [noexcept], [inherited]
```

Likewise, but not thread-safe.
References _M_attach_single().
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_attach_single(), and _M_attach_single().

_M_attached_to()

```
bool __gnu_debug::_Safe_iterator_base::_M_attached_to (
            const _Safe_sequence_base * __seq) const  [inline], [inherited]
```

Determines if we are attached to the given sequence.
References _M_attached_to(), and _M_sequence.
Referenced by _M_attached_to().

_M_before_dereferenceable()

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_M_before_dereferenceable ()
const  [inline]
```

Is the iterator before a dereferenceable one?

_M_can_compare()

```
bool __gnu_debug::_Safe_iterator_base::_M_can_compare (
            const _Safe_iterator_base & __x) const  [noexcept], [inherited]
```

Can we compare this iterator to the given iterator __x? Returns true if both iterators are nonsingular and reference the same sequence.
References _Safe_iterator_base(), and _M_can_compare().
Referenced by _M_can_compare().

_M_dereferenceable()

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_M_dereferenceable () const
[inline]
```

Is the iterator dereferenceable?
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator*(), and __gnu_debug::_Safe_iterator< _Base_iterator, m

_M_detach()

```
void __gnu_debug::_Safe_iterator_base::_M_detach () [protected], [inherited]
```

Detach the iterator for whatever sequence it is attached to, if any.
References _M_detach().
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), _M_detach(), __gnu_debug::_Safe_iterator< _Ba
and __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=().

_M_detach_single()

```
void __gnu_debug::_Safe_iterator_base::_M_detach_single () [noexcept], [inherited]
```

Likewise, but not thread-safe.

_M_get_mutex()

```
__gnu_cxx::__mutex & __gnu_debug::_Safe_iterator_base::_M_get_mutex () [protected], [noexcept],
[inherited]
```

For use in _Safe_iterator.
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator++(), __gnu_debug::_Safe_local_iterator< _OtherIterator,
__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(),
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=(), and __gnu_debug::_Safe_local_iterator< _OtherIterato

_M_incrementable()

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_M_incrementable () const
[inline]
```

Is the iterator incrementable?
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_before_dereferenceable(), __gnu_debug::_Safe_iterator< _Ba
and __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator++().

_M_invalidate()

```
void __gnu_debug::_Safe_iterator_base::_M_invalidate () [inline], [inherited]
```

Invalidate the iterator, making it singular.
References _M_invalidate(), and _M_version.
Referenced by _M_invalidate().

_M_is_before_begin()

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_M_is_before_begin () const
[inline]
```

Is this iterator equal to the sequence's before_begin() iterator if any?
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_dereferenceable().

_M_is_begin()

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_M_is_begin () const  [inline],
[constexpr]
```

Is this iterator equal to the sequence's begin() iterator?

_M_is_beginnest()

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_M_is_beginnest () const
[inline]
```

Is this iterator equal to the sequence's before_begin() iterator if any or begin() otherwise?

_M_is_end()

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_M_is_end () const  [inline]
```

Is this iterator equal to the sequence's end() iterator?

Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_dereferenceable(), and __gnu_debug::_Safe_iterator< _Base_

_M_reset()

```
void __gnu_debug::_Safe_iterator_base::_M_reset ()  [noexcept], [inherited]
```

Reset all member variables

_M_singular()

```
bool __gnu_debug::_Safe_iterator_base::_M_singular () const  [noexcept], [inherited]
```

Is this iterator singular?

Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), __gnu_debug::_Safe_iterator< _Base_iterator, m
__gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContaine
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_Safe_local_iterator(), __gnu_debug::_Safe_local_iterator< _Other
__gnu_debug::__check_singular_aux(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_dereferenceable(),
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_M_dereferenceable(), __gnu_debug::_Safe_iterator< _Base_itera
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_M_incrementable(), __gnu_debug::_Safe_iterator< _Base_iterator
__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(), __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >
and __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=().

_M_unlink()

```
void __gnu_debug::_Safe_iterator_base::_M_unlink ()  [inline], [noexcept], [inherited]
```

Unlink itself

References _M_next, _M_prior, and _M_unlink().

Referenced by _M_unlink().

_M_value_initialized()

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_M_value_initialized ()
const  [inline]
```

Is the iterator value-initialized?
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), __gnu_debug::_Safe_iterator< _Base_iterator, m
__gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator
and __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=().

_S_constant()

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
static constexpr bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_S_constant
() [inline], [static], [constexpr]
```

Determine if this is a constant iterator.
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_attach(), and __gnu_debug::_Safe_iterator< _Base_iterator, r

base()

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
_Iterator & __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::base () [inline],
[constexpr], [noexcept]
```

Return the underlying iterator.
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), __gnu_debug::_Safe_iterator< _Base_iterator, m
__gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_is_begin(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_is_en
__gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_value_initialized(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::op
__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator++(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator->
__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(), and __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator

operator _Iterator()

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::operator _Iterator () const [inline],
[constexpr], [noexcept]
```

Conversion to underlying non-debug iterator to allow better interaction with non-debug containers.

operator∗()

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
reference __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::operator* () const
[inline], [nodiscard], [constexpr], [noexcept]
```

Iterator dereference.

Precondition

iterator is dereferenceable

operator++() [1/2]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
_Safe_iterator & __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::operator++ ()
[inline], [constexpr], [noexcept]
```

Iterator preincrement.

Precondition

iterator is incrementable

operator++() [2/2]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
_Safe_iterator __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::operator++ (
            int )  [inline], [constexpr], [noexcept]
```

Iterator postincrement.

Precondition

iterator is incrementable

operator->()

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
pointer __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::operator-> () const  [inline],
[nodiscard], [constexpr], [noexcept]
```

Iterator dereference.

Precondition

iterator is dereferenceable

operator=() [1/2]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
_Safe_iterator & __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::operator= (
            _Safe_iterator< _Iterator, _Sequence, _Category > && __x)  [inline], [constexpr],
[noexcept]
```

Move assignment.

Postcondition

__x is singular and unattached

operator=() [2/2]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
_Safe_iterator & __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::operator= (
            const _Safe_iterator< _Iterator, _Sequence, _Category > & __x)  [inline], [constexpr],
[noexcept]
```

Copy assignment.

5.150.4 Member Data Documentation

_M_next

```
_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_next  [inherited]
```

Pointer to the next iterator in the sequence's list of iterators. Only valid when _M_sequence != NULL.

Referenced by _Safe_iterator_base(), _Safe_iterator_base(), _Safe_iterator_base(), __gnu_debug::_Safe_sequence< _SafeSequence >
__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if(), and _M_unlink().

_M_prior

_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_prior [inherited]
Pointer to the previous iterator in the sequence's list of iterators. Only valid when _M_sequence != NULL.
Referenced by _Safe_iterator_base(), _Safe_iterator_base(), _Safe_iterator_base(), __gnu_debug::_Safe_sequence< _SafeSequence > __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if(), and _M_unlink().

_M_sequence

const _Safe_sequence_base* __gnu_debug::_Safe_iterator_base::_M_sequence [inherited]
The sequence this iterator references; may be NULL to indicate a singular iterator. Stored as pointer-to-const because sequence could be declared as const.
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), __gnu_debug::_Safe_iterator< _Base_iterator, m __gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), _Safe_iterator_base(), _Safe_iterator_base(), _Safe_iterator_base(), _M_attached_to(), __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(), __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=(), and __gnu_debug::_Safe_local_iterator< _OtherIterato

_M_version

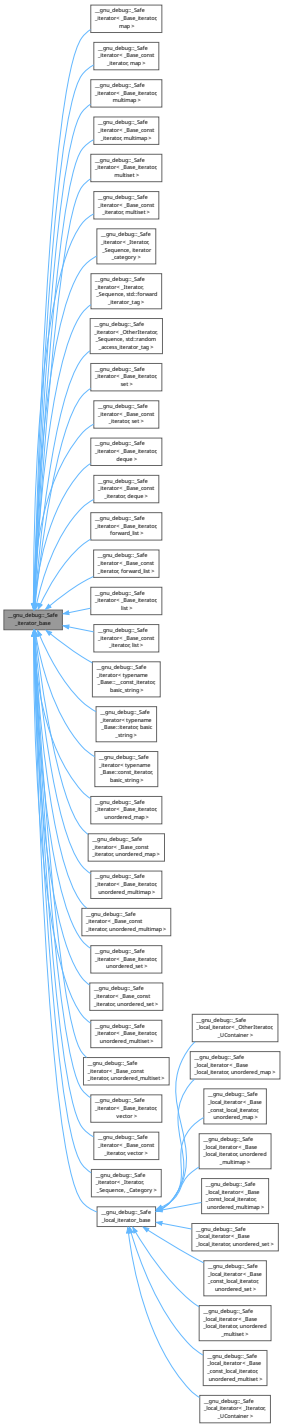
unsigned int __gnu_debug::_Safe_iterator_base::_M_version [inherited]
The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by _M_sequence for the iterator to be non-singular.
Referenced by _Safe_iterator_base(), _Safe_iterator_base(), _Safe_iterator_base(), _M_invalidate(), and __gnu_debug::_Safe_sequence
The documentation for this class was generated from the following files:

- boost_concept_check.h
- safe_iterator.h
- safe_iterator.tcc

5.151 __gnu_debug::_Safe_iterator_base Class Reference

#include <safe_base.h>

Inheritance diagram for __gnu_debug::_Safe_iterator_base:



Public Member Functions

- bool _M_attached_to (const _Safe_sequence_base *__seq) const
- bool _M_can_compare (const _Safe_iterator_base &__x) const noexcept

- void _M_detach_single () noexcept
- void _M_invalidate ()
- void _M_reset () noexcept
- bool _M_singular () const noexcept
- void _M_unlink () noexcept

Public Attributes

- _Safe_iterator_base * _M_next
- _Safe_iterator_base * _M_prior
- const _Safe_sequence_base * _M_sequence
- unsigned int _M_version

Protected Member Functions

- constexpr _Safe_iterator_base ()
- constexpr _Safe_iterator_base (const _Safe_iterator_base &__x, bool __constant)
- constexpr _Safe_iterator_base (const _Safe_sequence_base *__seq, bool __constant)
- void _M_attach (const _Safe_sequence_base *__seq, bool __constant)
- void _M_attach_single (const _Safe_sequence_base *__seq, bool __constant) noexcept
- void _M_detach ()
- __gnu_cxx::__mutex & _M_get_mutex () noexcept

Friends

- class **_Safe_sequence_base**

5.151.1 Detailed Description

Basic functionality for a *safe* iterator.
The _Safe_iterator_base base class implements the functionality of a safe iterator that is not specific to a particular iterator type. It contains a pointer back to the sequence it references along with iterator version information and pointers to form a doubly-linked list of iterators referenced by the container.
This class must not perform any operations that can throw an exception, or the exception guarantees of derived iterators will be broken.

5.151.2 Constructor & Destructor Documentation

_Safe_iterator_base() [1/3]

```
__gnu_debug::_Safe_iterator_base::_Safe_iterator_base ()  [inline], [constexpr], [protected]
```

Initializes the iterator and makes it singular.
References _M_next, _M_prior, _M_sequence, and _M_version.
Referenced by _Safe_iterator_base(), and _M_can_compare().

_Safe_iterator_base() [2/3]

```
__gnu_debug::_Safe_iterator_base::_Safe_iterator_base (
            const _Safe_sequence_base * __seq,
            bool __constant)  [inline], [constexpr], [protected]
```

Initialize the iterator to reference the sequence pointed to by __seq. __constant is true when we are initializing a constant iterator, and false if it is a mutable iterator. Note that __seq may be NULL, in which case the iterator will be singular. Otherwise, the iterator will reference __seq and be nonsingular.
References _M_attach(), _M_next, _M_prior, _M_sequence, and _M_version.

_Safe_iterator_base() [3/3]

```
__gnu_debug::_Safe_iterator_base::_Safe_iterator_base (
            const _Safe_iterator_base & __x,
            bool __constant)  [inline], [constexpr], [protected]
```

Initializes the iterator to reference the same sequence that __x does. __constant is true if this is a constant iterator, and false if it is mutable.
References _Safe_iterator_base(), _M_attach(), _M_next, _M_prior, _M_sequence, and _M_version.

5.151.3 Member Function Documentation

_M_attach()

```
void __gnu_debug::_Safe_iterator_base::_M_attach (
            const _Safe_sequence_base * __seq,
            bool __constant)  [protected]
```

Attaches this iterator to the given sequence, detaching it from whatever sequence it was attached to originally. If the new sequence is the NULL pointer, the iterator is left unattached.
References _M_attach().
Referenced by _Safe_iterator_base(), _Safe_iterator_base(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_attach(), and _M_attach().

_M_attach_single()

```
void __gnu_debug::_Safe_iterator_base::_M_attach_single (
            const _Safe_sequence_base * __seq,
            bool __constant)  [protected], [noexcept]
```

Likewise, but not thread-safe.
References _M_attach_single().
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_attach_single(), and _M_attach_single().

_M_attached_to()

```
bool __gnu_debug::_Safe_iterator_base::_M_attached_to (
            const _Safe_sequence_base * __seq) const  [inline]
```

Determines if we are attached to the given sequence.
References _M_attached_to(), and _M_sequence.
Referenced by _M_attached_to().

_M_can_compare()

```
bool __gnu_debug::_Safe_iterator_base::_M_can_compare (
            const _Safe_iterator_base & __x) const  [noexcept]
```

Can we compare this iterator to the given iterator __x? Returns true if both iterators are nonsingular and reference the same sequence.
References _Safe_iterator_base(), and _M_can_compare().
Referenced by _M_can_compare().

_M_detach()

```
void __gnu_debug::_Safe_iterator_base::_M_detach ()  [protected]
```

Detach the iterator for whatever sequence it is attached to, if any.
References _M_detach().
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), _M_detach(), __gnu_debug::_Safe_iterator< _Ba
and __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=().

_M_detach_single()

```
void __gnu_debug::_Safe_iterator_base::_M_detach_single ()  [noexcept]
```
Likewise, but not thread-safe.

_M_get_mutex()

```
__gnu_cxx::__mutex & __gnu_debug::_Safe_iterator_base::_M_get_mutex ()  [protected], [noexcept]
```
For use in _Safe_iterator.
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator++(), __gnu_debug::_Safe_local_iterator< _OtherIterator,
__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(),
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=(), and __gnu_debug::_Safe_local_iterator< _OtherIterato

_M_invalidate()

```
void __gnu_debug::_Safe_iterator_base::_M_invalidate ()  [inline]
```
Invalidate the iterator, making it singular.
References _M_invalidate(), and _M_version.
Referenced by _M_invalidate().

_M_reset()

```
void __gnu_debug::_Safe_iterator_base::_M_reset ()  [noexcept]
```
Reset all member variables

_M_singular()

```
bool __gnu_debug::_Safe_iterator_base::_M_singular () const  [noexcept]
```
Is this iterator singular?
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), __gnu_debug::_Safe_iterator< _Base_iterator, m
__gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContaine
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_Safe_local_iterator(), __gnu_debug::_Safe_local_iterator< _Other
__gnu_debug::__check_singular_aux(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_dereferenceable(),
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_M_dereferenceable(), __gnu_debug::_Safe_iterator< _Base_itera
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_M_incrementable(), __gnu_debug::_Safe_iterator< _Base_iterato
__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(), __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >
and __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=().

_M_unlink()

```
void __gnu_debug::_Safe_iterator_base::_M_unlink ()  [inline], [noexcept]
```
Unlink itself
References _M_next, _M_prior, and _M_unlink().
Referenced by _M_unlink().

5.151.4 Member Data Documentation

_M_next

```
_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_next
```
Pointer to the next iterator in the sequence's list of iterators. Only valid when _M_sequence != NULL.
Referenced by _Safe_iterator_base(), _Safe_iterator_base(), _Safe_iterator_base(), __gnu_debug::_Safe_sequence< _SafeSequence >
__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if(), and _M_unlink().

_M_prior

_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_prior

Pointer to the previous iterator in the sequence's list of iterators. Only valid when _M_sequence != NULL.

Referenced by _Safe_iterator_base(), _Safe_iterator_base(), _Safe_iterator_base(), __gnu_debug::_Safe_sequence< _SafeSequence > __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if(), and _M_unlink().

_M_sequence

const _Safe_sequence_base* __gnu_debug::_Safe_iterator_base::_M_sequence

The sequence this iterator references; may be NULL to indicate a singular iterator. Stored as pointer-to-const because sequence could be declared as const.

Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), __gnu_debug::_Safe_iterator< _Base_iterator, m __gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), _Safe_iterator_base(), _Safe_iterator_base(), _Safe_iterator_base(), _M_attached_to(), __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(), __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=(), and __gnu_debug::_Safe_local_iterator< _OtherIterato

_M_version

unsigned int __gnu_debug::_Safe_iterator_base::_M_version

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by _M_sequence for the iterator to be non-singular.

Referenced by _Safe_iterator_base(), _Safe_iterator_base(), _Safe_iterator_base(), _M_invalidate(), and __gnu_debug::_Safe_sequence

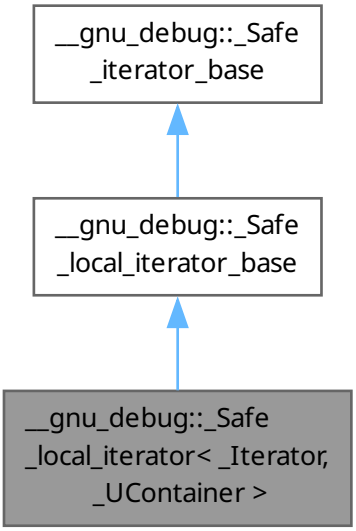
The documentation for this class was generated from the following file:

- safe_base.h

5.152 __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer > Class Template Reference

#include <safe_local_iterator.h>

Inheritance diagram for __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >:



Public Types

- typedef _Traits::difference_type **difference_type**
- typedef _Traits::iterator_category **iterator_category**
- typedef _Iterator **iterator_type**
- typedef _Traits::pointer **pointer**
- typedef _Traits::reference **reference**
- typedef _Traits::value_type **value_type**

Public Member Functions

- _Safe_local_iterator () noexcept
- _Safe_local_iterator (_Iterator __i, const _Safe_unordered_container_base *__cont)
- _Safe_local_iterator (_Safe_local_iterator &&__x) noexcept
- _Safe_local_iterator (const _Safe_local_iterator &__x) noexcept
- template<typename _MutableIterator>
  _Safe_local_iterator (const _Safe_local_iterator< _MutableIterator, typename __gnu_cxx::__enable_if< _IsConstant::__value &&std::__are_same< _MutableIterator, _OtherIterator >::__value, _UContainer >::__type > &__x) noexcept
- void _M_attach (const _Safe_unordered_container_base *__cont)
- void _M_attach_single (const _Safe_unordered_container_base *__cont)
- bool _M_attached_to (const _Safe_sequence_base *__seq) const
- bool _M_can_compare (const _Safe_iterator_base &__x) const noexcept
- bool _M_dereferenceable () const
- _Distance_traits< _Iterator >::__type **_M_get_distance_to** (const _Safe_local_iterator &__rhs) const
- std::__conditional_t< _IsConstant::__value, const _UContainer *, _UContainer * > **_M_get_ucontainer** () const

- template<typename _Other>
  bool _M_in_same_bucket (const _Safe_local_iterator< _Other, _UContainer > &__other) const
- bool _M_incrementable () const
- void _M_invalidate ()
- bool _M_is_begin () const
- bool _M_is_end () const
- void _M_reset () noexcept
- const _Safe_unordered_container_base * **_M_safe_container** () const noexcept
- bool _M_singular () const noexcept
- void _M_unlink () noexcept
- bool **_M_valid_range** (const _Safe_local_iterator &__rhs, std::pair< difference_type, _Distance_precision > &↩
  __dist_info) const
- bool _M_value_initialized () const
- const _Iterator & **base** () const noexcept
- _Iterator & base () noexcept
- size_type bucket () const
- operator _Iterator () const
- reference operator* () const
- _Safe_local_iterator & operator++ ()
- _Safe_local_iterator operator++ (int)
- pointer operator-> () const
- _Safe_local_iterator & operator= (_Safe_local_iterator &&__x) noexcept
- _Safe_local_iterator & operator= (const _Safe_local_iterator &__x)

Static Public Member Functions

- static constexpr bool _S_constant ()

Public Attributes

- _Safe_iterator_base * _M_next
- _Safe_iterator_base * _M_prior
- const _Safe_sequence_base * _M_sequence
- unsigned int _M_version

Protected Member Functions

- void _M_attach (const _Safe_sequence_base *__seq, bool __constant)
- void _M_attach (const _Safe_unordered_container_base *__cont, bool __constant)
- void _M_attach_single (const _Safe_sequence_base *__seq, bool __constant) noexcept
- void _M_attach_single (const _Safe_unordered_container_base *__cont, bool __constant) noexcept
- void _M_detach ()
- void _M_detach_single () noexcept
- __gnu_cxx::__mutex & _M_get_mutex () noexcept

Friends

- bool **operator!=** (const _Self &__lhs, const _OtherSelf &__rhs) noexcept
- bool **operator!=** (const _Self &__lhs, const _Self &__rhs) noexcept
- bool **operator==** (const _Self &__lhs, const _OtherSelf &__rhs) noexcept
- bool **operator==** (const _Self &__lhs, const _Self &__rhs) noexcept

5.152.1 Detailed Description

template<typename _Iterator, typename _UContainer>
class __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >

Safe iterator wrapper.
The class template _Safe_local_iterator is a wrapper around an iterator that tracks the iterator's movement among unordered containers and checks that operations performed on the "safe" iterator are legal. In additional to the basic iterator operations (which are validated, and then passed to the underlying iterator), _Safe_local_iterator has member functions for iterator invalidation, attaching/detaching the iterator from unordered containers, and querying the iterator's state.

5.152.2 Constructor & Destructor Documentation

_Safe_local_iterator() [1/5]

```
template<typename _Iterator, typename _UContainer>
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::_Safe_local_iterator ()  [inline],
[noexcept]
```

Postcondition

the iterator is singular and unattached

_Safe_local_iterator() [2/5]

```
template<typename _Iterator, typename _UContainer>
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::_Safe_local_iterator (
            _Iterator __i,
            const _Safe_unordered_container_base * __cont)  [inline]
```

Safe iterator construction from an unsafe iterator and its unordered container.

Precondition

`cont` is not NULL

Postcondition

this is not singular

_Safe_local_iterator() [3/5]

```
template<typename _Iterator, typename _UContainer>
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::_Safe_local_iterator (
            const _Safe_local_iterator< _Iterator, _UContainer > & __x)  [inline], [noexcept]
```

Copy construction.

_Safe_local_iterator() [4/5]

```
template<typename _Iterator, typename _UContainer>
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::_Safe_local_iterator (
            _Safe_local_iterator< _Iterator, _UContainer > && __x)  [inline], [noexcept]
```

Move construction.

Postcondition

__x is singular and unattached

_Safe_local_iterator() [5/5]

```
template<typename _Iterator, typename _UContainer>
template<typename _MutableIterator>
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::_Safe_local_iterator (
            const _Safe_local_iterator< _MutableIterator, typename __gnu_cxx::__enable_if< _↩
IsConstant::__value &&std::__are_same< _MutableIterator, _OtherIterator >::__value, _UContainer
>::__type > & __x)  [inline], [noexcept]
```

Converting constructor from a mutable iterator to a constant iterator.

5.152.3 Member Function Documentation

_M_attach() [1/3]

```
void __gnu_debug::_Safe_iterator_base::_M_attach (
            const _Safe_sequence_base * __seq,
            bool __constant)  [protected], [inherited]
```

Attaches this iterator to the given sequence, detaching it from whatever sequence it was attached to originally. If the new sequence is the NULL pointer, the iterator is left unattached.

References _M_attach().

Referenced by _Safe_iterator_base(), _Safe_iterator_base(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_attach(), and _M_attach().

_M_attach() [2/3]

```
template<typename _Iterator, typename _UContainer>
void __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::_M_attach (
            const _Safe_unordered_container_base * __cont)  [inline]
```

Attach iterator to the given unordered container.

Referenced by __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_Safe_local_iterator(), __gnu_debug::_Safe_local_it
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_Safe_local_iterator(), __gnu_debug::_Safe_local_iterator< _Other
and __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=().

_M_attach() [3/3]

```
void __gnu_debug::_Safe_local_iterator_base::_M_attach (
            const _Safe_unordered_container_base * __cont,
            bool __constant)  [protected], [inherited]
```

Attaches this iterator to the given container, detaching it from whatever container it was attached to originally. If the new container is the NULL pointer, the iterator is left unattached.

Referenced by _Safe_local_iterator_base(), _Safe_local_iterator_base(), and __gnu_debug::_Safe_local_iterator< _OtherIterator, _UCon

_M_attach_single() [1/3]

```
void __gnu_debug::_Safe_iterator_base::_M_attach_single (
            const _Safe_sequence_base * __seq,
            bool __constant)  [protected], [noexcept], [inherited]
```

Likewise, but not thread-safe.

References _M_attach_single().

Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_attach_single(), and _M_attach_single().

_M_attach_single() [2/3]

```
template<typename _Iterator, typename _UContainer>
void __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::_M_attach_single (
            const _Safe_unordered_container_base * __cont)  [inline]
```

Likewise, but not thread-safe.

_M_attach_single() [3/3]

```
void __gnu_debug::_Safe_local_iterator_base::_M_attach_single (
            const _Safe_unordered_container_base * __cont,
            bool __constant)  [protected], [noexcept], [inherited]
```

Likewise, but not thread-safe.
Referenced by __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_M_attach_single().

_M_attached_to()

```
bool __gnu_debug::_Safe_iterator_base::_M_attached_to (
            const _Safe_sequence_base * __seq) const  [inline], [inherited]
```

Determines if we are attached to the given sequence.
References _M_attached_to(), and _M_sequence.
Referenced by _M_attached_to().

_M_can_compare()

```
bool __gnu_debug::_Safe_iterator_base::_M_can_compare (
            const _Safe_iterator_base & __x) const  [noexcept], [inherited]
```

Can we compare this iterator to the given iterator __x? Returns true if both iterators are nonsingular and reference the same sequence.
References _Safe_iterator_base(), and _M_can_compare().
Referenced by _M_can_compare().

_M_dereferenceable()

```
template<typename _Iterator, typename _UContainer>
bool __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::_M_dereferenceable () const
[inline]
```

Is the iterator dereferenceable?
Referenced by __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator*(), and __gnu_debug::_Safe_local_iterator<

_M_detach()

```
void __gnu_debug::_Safe_local_iterator_base::_M_detach ()  [protected], [inherited]
```

Detach the iterator for whatever container it is attached to, if any.
Referenced by __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_Safe_local_iterator(), __gnu_debug::_Safe_local_it
and __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=().

_M_detach_single()

```
void __gnu_debug::_Safe_local_iterator_base::_M_detach_single ()  [protected], [noexcept], [inherited]
```

Likewise, but not thread-safe.

_M_get_mutex()

```
__gnu_cxx::__mutex & __gnu_debug::_Safe_iterator_base::_M_get_mutex ()  [protected], [noexcept],
[inherited]
```

For use in _Safe_iterator.
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator++(), __gnu_debug::_Safe_local_iterator< _OtherIterator,
__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(),
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=(), and __gnu_debug::_Safe_local_iterator< _OtherIterato

_M_in_same_bucket()

```
template<typename _Iterator, typename _UContainer>
template<typename _Other>
bool __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::_M_in_same_bucket (
             const _Safe_local_iterator< _Other, _UContainer > & __other) const  [inline]
```

Is this iterator part of the same bucket as the other one?

_M_incrementable()

```
template<typename _Iterator, typename _UContainer>
bool __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::_M_incrementable () const  [inline]
```

Is the iterator incrementable?
Referenced by __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator++(), and __gnu_debug::_Safe_local_iterato

_M_invalidate()

```
void __gnu_debug::_Safe_iterator_base::_M_invalidate ()  [inline], [inherited]
```

Invalidate the iterator, making it singular.
References _M_invalidate(), and _M_version.
Referenced by _M_invalidate().

_M_is_begin()

```
template<typename _Iterator, typename _UContainer>
bool __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::_M_is_begin () const  [inline]
```

Is this iterator equal to the container's begin(bucket) iterator?

_M_is_end()

```
template<typename _Iterator, typename _UContainer>
bool __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::_M_is_end () const  [inline]
```

Is this iterator equal to the container's end(bucket) iterator?
Referenced by __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_M_dereferenceable(), and __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_M_incrementable().

_M_reset()

```
void __gnu_debug::_Safe_iterator_base::_M_reset ()  [noexcept], [inherited]
```

Reset all member variables

_M_singular()

```
bool __gnu_debug::_Safe_iterator_base::_M_singular () const  [noexcept], [inherited]
```

Is this iterator singular?
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), __gnu_debug::_Safe_iterator< _Base_iterator, m
__gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContaine
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_Safe_local_iterator(), __gnu_debug::_Safe_local_iterator< _Other
__gnu_debug::__check_singular_aux(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_dereferenceable(),
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_M_dereferenceable(), __gnu_debug::_Safe_iterator< _Base_itera
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_M_incrementable(), __gnu_debug::_Safe_iterator< _Base_iterator
__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(), __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >
and __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=().

_M_unlink()

```
void __gnu_debug::_Safe_iterator_base::_M_unlink ()  [inline], [noexcept], [inherited]
```

Unlink itself

References _M_next, _M_prior, and _M_unlink().

Referenced by _M_unlink().

_M_value_initialized()

```
template<typename _Iterator, typename _UContainer>
bool __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::_M_value_initialized () const
[inline]
```

Is the iterator value-initialized?

Referenced by __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_Safe_local_iterator(), __gnu_debug::_Safe_local_ite
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_Safe_local_iterator(), __gnu_debug::_Safe_local_iterator< _Other
and __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=().

_S_constant()

```
template<typename _Iterator, typename _UContainer>
static constexpr bool __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::_S_constant ()
[inline], [static], [constexpr]
```

Determine if this is a constant iterator.

Referenced by __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_M_attach(), and __gnu_debug::_Safe_local_iterator

base()

```
template<typename _Iterator, typename _UContainer>
_Iterator & __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::base ()  [inline], [noexcept]
```

Return the underlying iterator.

Referenced by __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_Safe_local_iterator(), __gnu_debug::_Safe_local_ite
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_M_is_end(), __gnu_debug::_Safe_local_iterator< _OtherIterator, _
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::bucket(), __gnu_debug::_Safe_local_iterator< _OtherIterator, _UCo
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator++(), __gnu_debug::_Safe_local_iterator< _OtherIterator, _
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=(), and __gnu_debug::_Safe_local_iterator< _OtherIterato

bucket()

```
template<typename _Iterator, typename _UContainer>
size_type __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::bucket () const  [inline]
```

Return the bucket.

Referenced by __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_M_in_same_bucket(), __gnu_debug::_Safe_local_it
and __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_M_is_end().

operator _Iterator()

```
template<typename _Iterator, typename _UContainer>
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::operator _Iterator () const  [inline]
```

Conversion to underlying non-debug iterator to allow better interaction with non-debug containers.

operator*()

```
template<typename _Iterator, typename _UContainer>
reference __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::operator* () const  [inline]
```

Iterator dereference.

Precondition

iterator is dereferenceable

operator++() [1/2]

```
template<typename _Iterator, typename _UContainer>
_Safe_local_iterator & __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::operator++ ()
[inline]
```

Iterator preincrement.

Precondition

iterator is incrementable

operator++() [2/2]

```
template<typename _Iterator, typename _UContainer>
_Safe_local_iterator __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::operator++ (
            int )  [inline]
```

Iterator postincrement.

Precondition

iterator is incrementable

operator->()

```
template<typename _Iterator, typename _UContainer>
pointer __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::operator-> () const  [inline]
```

Iterator dereference.

Precondition

iterator is dereferenceable

operator=() [1/2]

```
template<typename _Iterator, typename _UContainer>
_Safe_local_iterator & __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::operator= (
            _Safe_local_iterator< _Iterator, _UContainer > && __x)  [inline], [noexcept]
```

Move assignment.

Postcondition

__x is singular and unattached

operator=() [2/2]

```
template<typename _Iterator, typename _UContainer>
_Safe_local_iterator & __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::operator= (
            const _Safe_local_iterator< _Iterator, _UContainer > & __x)  [inline]
```

Copy assignment.

5.152.4 Member Data Documentation

_M_next

_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_next [inherited]
Pointer to the next iterator in the sequence's list of iterators. Only valid when _M_sequence != NULL.
Referenced by _Safe_iterator_base(), _Safe_iterator_base(), _Safe_iterator_base(), __gnu_debug::_Safe_sequence< _SafeSequence >
__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if(), and _M_unlink().

_M_prior

_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_prior [inherited]
Pointer to the previous iterator in the sequence's list of iterators. Only valid when _M_sequence != NULL.
Referenced by _Safe_iterator_base(), _Safe_iterator_base(), _Safe_iterator_base(), __gnu_debug::_Safe_sequence< _SafeSequence >
__gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if(), and _M_unlink().

_M_sequence

const _Safe_sequence_base* __gnu_debug::_Safe_iterator_base::_M_sequence [inherited]
The sequence this iterator references; may be NULL to indicate a singular iterator. Stored as pointer-to-const because sequence could be declared as const.
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), __gnu_debug::_Safe_iterator< _Base_iterator, m
__gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), _Safe_iterator_base(), _Safe_iterator_base(),
_Safe_iterator_base(), _M_attached_to(), __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if(),
__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(),
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=(), and __gnu_debug::_Safe_local_iterator< _OtherIterato

_M_version

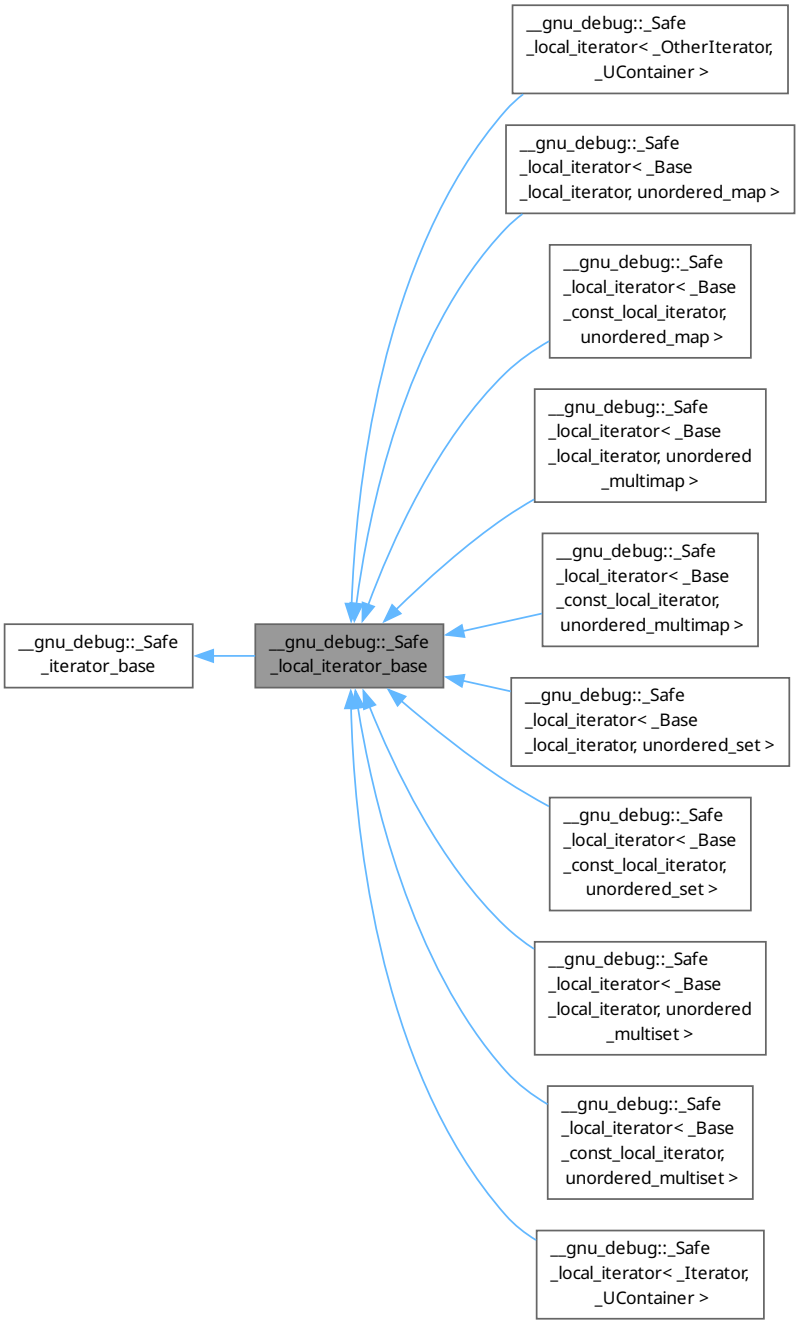
unsigned int __gnu_debug::_Safe_iterator_base::_M_version [inherited]
The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by _M_sequence for the iterator to be non-singular.
Referenced by _Safe_iterator_base(), _Safe_iterator_base(), _Safe_iterator_base(), _M_invalidate(), and __gnu_debug::_Safe_sequence
The documentation for this class was generated from the following files:

- formatter.h
- safe_local_iterator.h
- safe_local_iterator.tcc

5.153 __gnu_debug::_Safe_local_iterator_base Class Reference

#include <safe_unordered_base.h>

Inheritance diagram for __gnu_debug::_Safe_local_iterator_base:



Public Member Functions

- bool _M_attached_to (const _Safe_sequence_base *__seq) const
- bool _M_can_compare (const _Safe_iterator_base &__x) const noexcept
- void _M_invalidate ()

- void _M_reset () noexcept
- const _Safe_unordered_container_base * **_M_safe_container** () const noexcept
- bool _M_singular () const noexcept
- void _M_unlink () noexcept

Public Attributes

- _Safe_iterator_base * _M_next
- _Safe_iterator_base * _M_prior
- const _Safe_sequence_base * _M_sequence
- unsigned int _M_version

Protected Member Functions

- _Safe_local_iterator_base ()
- _Safe_local_iterator_base (const _Safe_local_iterator_base &__x, bool __constant)
- _Safe_local_iterator_base (const _Safe_unordered_container_base *__seq, bool __constant)
- void _M_attach (const _Safe_sequence_base *__seq, bool __constant)
- void _M_attach (const _Safe_unordered_container_base *__cont, bool __constant)
- void _M_attach_single (const _Safe_sequence_base *__seq, bool __constant) noexcept
- void _M_attach_single (const _Safe_unordered_container_base *__cont, bool __constant) noexcept
- void _M_detach ()
- void _M_detach_single () noexcept
- __gnu_cxx::__mutex & _M_get_mutex () noexcept

5.153.1 Detailed Description

Basic functionality for a *safe* iterator.
The _Safe_local_iterator_base base class implements the functionality of a safe local iterator that is not specific to a particular iterator type. It contains a pointer back to the container it references along with iterator version information and pointers to form a doubly-linked list of local iterators referenced by the container.
This class must not perform any operations that can throw an exception, or the exception guarantees of derived iterators will be broken.

5.153.2 Constructor & Destructor Documentation

_Safe_local_iterator_base() [1/3]

```
__gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base ()  [inline], [protected]
```

Initializes the iterator and makes it singular.
Referenced by _Safe_local_iterator_base().

_Safe_local_iterator_base() [2/3]

```
__gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base (
            const _Safe_unordered_container_base * __seq,
            bool __constant)  [inline], [protected]
```

Initialize the iterator to reference the container pointed to by __seq. __constant is true when we are initializing a constant local iterator, and false if it is a mutable local iterator. Note that __seq may be NULL, in which case the iterator will be singular. Otherwise, the iterator will reference __seq and be nonsingular.
References _M_attach().

_Safe_local_iterator_base() [3/3]

```
__gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base (
            const _Safe_local_iterator_base & __x,
            bool __constant)  [inline], [protected]
```

Initializes the iterator to reference the same container that __x does. __constant is true if this is a constant iterator, and false if it is mutable.
References _Safe_local_iterator_base(), and _M_attach().

5.153.3 Member Function Documentation

_M_attach() [1/2]

```
void __gnu_debug::_Safe_iterator_base::_M_attach (
            const _Safe_sequence_base * __seq,
            bool __constant)  [protected], [inherited]
```

Attaches this iterator to the given sequence, detaching it from whatever sequence it was attached to originally. If the new sequence is the NULL pointer, the iterator is left unattached.
References _M_attach().
Referenced by _Safe_iterator_base(), _Safe_iterator_base(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_attach(), and _M_attach().

_M_attach() [2/2]

```
void __gnu_debug::_Safe_local_iterator_base::_M_attach (
            const _Safe_unordered_container_base * __cont,
            bool __constant)  [protected]
```

Attaches this iterator to the given container, detaching it from whatever container it was attached to originally. If the new container is the NULL pointer, the iterator is left unattached.
Referenced by _Safe_local_iterator_base(), _Safe_local_iterator_base(), and __gnu_debug::_Safe_local_iterator< _OtherIterator, _UCon

_M_attach_single() [1/2]

```
void __gnu_debug::_Safe_iterator_base::_M_attach_single (
            const _Safe_sequence_base * __seq,
            bool __constant)  [protected], [noexcept], [inherited]
```

Likewise, but not thread-safe.
References _M_attach_single().
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_attach_single(), and _M_attach_single().

_M_attach_single() [2/2]

```
void __gnu_debug::_Safe_local_iterator_base::_M_attach_single (
            const _Safe_unordered_container_base * __cont,
            bool __constant)  [protected], [noexcept]
```

Likewise, but not thread-safe.
Referenced by __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_M_attach_single().

_M_attached_to()

```
bool __gnu_debug::_Safe_iterator_base::_M_attached_to (
            const _Safe_sequence_base * __seq) const  [inline], [inherited]
```

Determines if we are attached to the given sequence.
References _M_attached_to(), and _M_sequence.
Referenced by _M_attached_to().

_M_can_compare()

```
bool __gnu_debug::_Safe_iterator_base::_M_can_compare (
            const _Safe_iterator_base & __x) const  [noexcept], [inherited]
```

Can we compare this iterator to the given iterator __x? Returns true if both iterators are nonsingular and reference the same sequence.
References _Safe_iterator_base(), and _M_can_compare().
Referenced by _M_can_compare().

_M_detach()

```
void __gnu_debug::_Safe_local_iterator_base::_M_detach ()  [protected]
```

Detach the iterator for whatever container it is attached to, if any.
Referenced by __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_Safe_local_iterator(), __gnu_debug::_Safe_local_ite
and __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=().

_M_detach_single()

```
void __gnu_debug::_Safe_local_iterator_base::_M_detach_single ()  [protected], [noexcept]
```

Likewise, but not thread-safe.

_M_get_mutex()

```
__gnu_cxx::__mutex & __gnu_debug::_Safe_iterator_base::_M_get_mutex ()  [protected], [noexcept],
[inherited]
```

For use in _Safe_iterator.
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator++(), __gnu_debug::_Safe_local_iterator< _OtherIterator,
__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(),
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=(), and __gnu_debug::_Safe_local_iterator< _OtherIterato

_M_invalidate()

```
void __gnu_debug::_Safe_iterator_base::_M_invalidate ()  [inline], [inherited]
```

Invalidate the iterator, making it singular.
References _M_invalidate(), and _M_version.
Referenced by _M_invalidate().

_M_reset()

```
void __gnu_debug::_Safe_iterator_base::_M_reset ()  [noexcept], [inherited]
```

Reset all member variables

_M_singular()

```
bool __gnu_debug::_Safe_iterator_base::_M_singular () const  [noexcept], [inherited]
```

Is this iterator singular?
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), __gnu_debug::_Safe_iterator< _Base_iterator, m
__gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContaine
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_Safe_local_iterator(), __gnu_debug::_Safe_local_iterator< _Other
__gnu_debug::__check_singular_aux(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_dereferenceable(),
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_M_dereferenceable(), __gnu_debug::_Safe_iterator< _Base_itera
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_M_incrementable(), __gnu_debug::_Safe_iterator< _Base_iterator
__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(), __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >
and __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=().

_M_unlink()

```
void __gnu_debug::_Safe_iterator_base::_M_unlink () [inline], [noexcept], [inherited]
```

Unlink itself

References _M_next, _M_prior, and _M_unlink().

Referenced by _M_unlink().

5.153.4 Member Data Documentation

_M_next

```
_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_next [inherited]
```

Pointer to the next iterator in the sequence's list of iterators. Only valid when _M_sequence != NULL.

Referenced by _Safe_iterator_base(), _Safe_iterator_base(), _Safe_iterator_base(), __gnu_debug::_Safe_sequence< _SafeSequence > __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if(), and _M_unlink().

_M_prior

```
_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_prior [inherited]
```

Pointer to the previous iterator in the sequence's list of iterators. Only valid when _M_sequence != NULL.

Referenced by _Safe_iterator_base(), _Safe_iterator_base(), _Safe_iterator_base(), __gnu_debug::_Safe_sequence< _SafeSequence > __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if(), and _M_unlink().

_M_sequence

```
const _Safe_sequence_base* __gnu_debug::_Safe_iterator_base::_M_sequence [inherited]
```

The sequence this iterator references; may be NULL to indicate a singular iterator. Stored as pointer-to-const because sequence could be declared as const.

Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), __gnu_debug::_Safe_iterator< _Base_iterator, m __gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), _Safe_iterator_base(), _Safe_iterator_base(), _Safe_iterator_base(), _M_attached_to(), __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(), __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=(), and __gnu_debug::_Safe_local_iterator< _OtherIterato

_M_version

```
unsigned int __gnu_debug::_Safe_iterator_base::_M_version [inherited]
```

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by _M_sequence for the iterator to be non-singular.

Referenced by _Safe_iterator_base(), _Safe_iterator_base(), _Safe_iterator_base(), _M_invalidate(), and __gnu_debug::_Safe_sequence

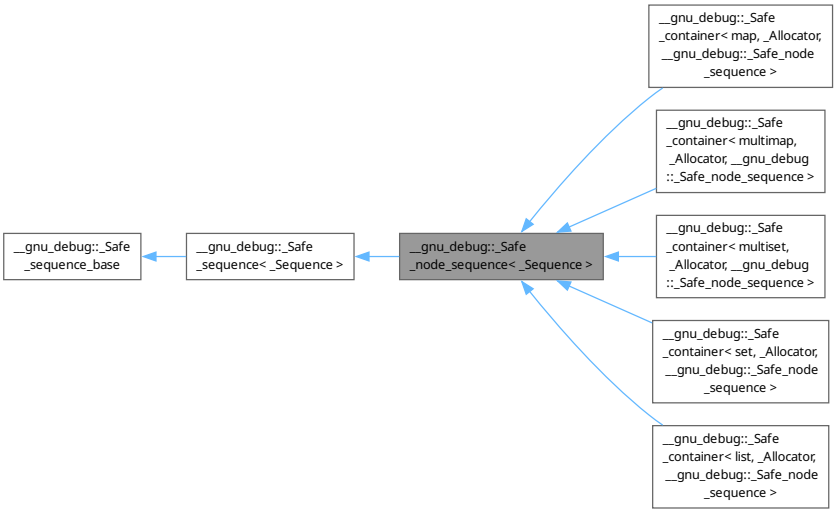
The documentation for this class was generated from the following file:

- safe_unordered_base.h

5.154 __gnu_debug::_Safe_node_sequence< _Sequence > Class Template Reference

```
#include <safe_sequence.h>
```

Inheritance diagram for __gnu_debug::_Safe_node_sequence< _Sequence >:



Public Member Functions

- template<typename _Predicate>
  void _M_invalidate_if (_Predicate __pred) const
- template<typename _Predicate>
  void _M_transfer_from_if (const _Safe_sequence &__from, _Predicate __pred) const

Public Attributes

- _Safe_iterator_base * _M_const_iterators
- _Safe_iterator_base * _M_iterators
- unsigned int _M_version

Protected Member Functions

- void _M_detach_all () const
- void _M_detach_singular () const
- __gnu_cxx::__mutex & _M_get_mutex () const noexcept
- void **_M_invalidate_all** () const
- void _M_revalidate_singular () const
- void _M_swap (const _Safe_sequence_base &__x) const noexcept

5.154.1 Detailed Description

template<typename _Sequence>
class __gnu_debug::_Safe_node_sequence< _Sequence >

Like _Safe_sequence but with a special _M_invalidate_all implementation not invalidating past-the-end iterators. Used by node based sequence.

5.154.2 Member Function Documentation

_M_detach_all()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_all () const  [protected], [inherited]
```

Detach all iterators, leaving them singular.
Referenced by ~_Safe_sequence_base().

_M_detach_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular () const  [protected], [inherited]
```

Detach all singular iterators.

Postcondition

for all iterators i attached to this sequence, i->_M_version == _M_version.

_M_get_mutex()

```
__gnu_cxx::__mutex & __gnu_debug::_Safe_sequence_base::_M_get_mutex () const  [protected], [noexcept],
[inherited]
```

For use in _Safe_sequence.
Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if(), __gnu_debug::_Safe_unordered_container< _Conta
__gnu_debug::_Safe_unordered_container< _Container >::_M_invalidate_local_if(), and __gnu_debug::_Safe_sequence< _Sequence

_M_invalidate_if()

```
template<typename _Sequence>
template<typename _Predicate>
void __gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if (
            _Predicate __pred) const  [inherited]
```

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal iterators nested in the safe ones.
References __gnu_debug::_Safe_sequence_base::_M_const_iterators, __gnu_debug::_Safe_sequence_base::_M_get_mutex(), and __gnu_debug::_Safe_sequence_base::_M_iterators.

_M_revalidate_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () const  [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

_M_swap()

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
            const _Safe_sequence_base & __x) const  [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

_M_transfer_from_if()

```
template<typename _Sequence>
template<typename _Predicate>
void __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if (
            const _Safe_sequence< _Sequence > & __from,
            _Predicate __pred) const  [inherited]
```

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal iterators nested in the safe ones.
References std::__addressof(), __gnu_debug::_Safe_sequence_base::_M_const_iterators, __gnu_debug::_Safe_sequence_base::_M_ge __gnu_debug::_Safe_sequence_base::_M_iterators, __gnu_debug::_Safe_iterator_base::_M_next, __gnu_debug::_Safe_iterator_base::_ __gnu_debug::_Safe_iterator_base::_M_sequence, __gnu_debug::_Safe_iterator_base::_M_version, and __gnu_debug::_Safe_sequenc

5.154.3 Member Data Documentation

_M_const_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators  [mutable], [inherited]
```

The list of constant iterators that reference this container.
Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if(), __gnu_debug::_Safe_unordered_container< _Conta and __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

_M_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators  [mutable], [inherited]
```

The list of mutable iterators that reference this container.
Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if(), __gnu_debug::_Safe_unordered_container< _Conta and __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

_M_version

```
unsigned int __gnu_debug::_Safe_sequence_base::_M_version  [mutable], [inherited]
```

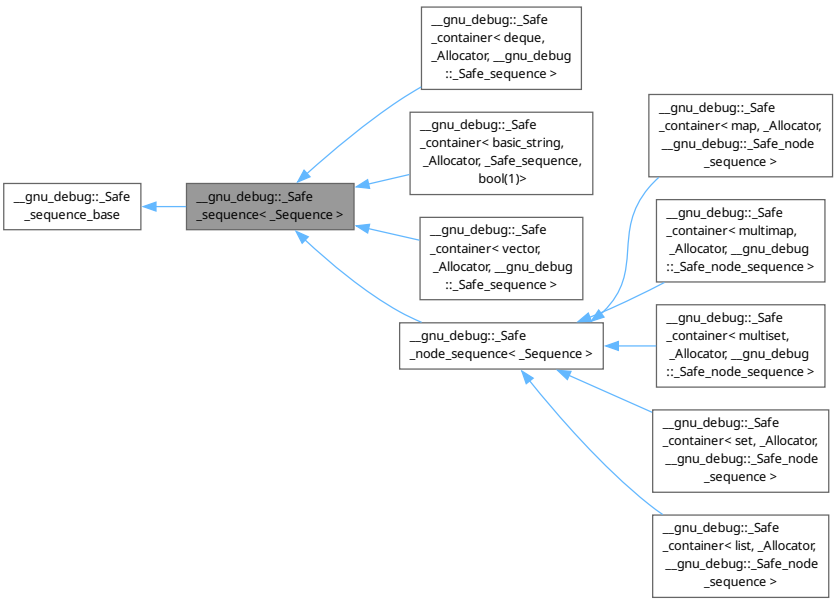
The container version number. This number may never be 0.
Referenced by _M_invalidate_all(), __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if(), __gnu_debug::_Safe_iterator __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(), __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer > and __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=().
The documentation for this class was generated from the following file:

- safe_sequence.h

5.155 __gnu_debug::_Safe_sequence< _Sequence > Class Template Reference

```
#include <safe_sequence.h>
```

Inheritance diagram for __gnu_debug::_Safe_sequence< _Sequence >:



Public Member Functions

- template<typename _Predicate>
  void _M_invalidate_if (_Predicate __pred) const
- template<typename _Predicate>
  void _M_transfer_from_if (const _Safe_sequence &__from, _Predicate __pred) const

Public Attributes

- _Safe_iterator_base * _M_const_iterators
- _Safe_iterator_base * _M_iterators
- unsigned int _M_version

Protected Member Functions

- void _M_detach_all () const
- void _M_detach_singular () const
- __gnu_cxx::__mutex & _M_get_mutex () const noexcept
- void _M_invalidate_all () const
- void _M_revalidate_singular () const
- void _M_swap (const _Safe_sequence_base &__x) const noexcept

5.155.1 Detailed Description

template<typename _Sequence>
class __gnu_debug::_Safe_sequence< _Sequence >

Base class for constructing a *safe* sequence type that tracks iterators that reference it.

The class template _Safe_sequence simplifies the construction of *safe* sequences that track the iterators that reference the sequence, so that the iterators are notified of changes in the sequence that may affect their operation, e.g., if the container invalidates its iterators or is destructed. This class template may only be used by deriving from it and passing the name of the derived class as its template parameter via the curiously recurring template pattern. The derived class must have `iterator` and `const_iterator` types that are instantiations of class template _Safe_iterator for this sequence. Iterators will then be tracked automatically.

5.155.2 Member Function Documentation

_M_detach_all()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_all () const  [protected], [inherited]
```

Detach all iterators, leaving them singular.
Referenced by ~_Safe_sequence_base().

_M_detach_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular () const  [protected], [inherited]
```

Detach all singular iterators.

Postcondition

for all iterators i attached to this sequence, i->_M_version == _M_version.

_M_get_mutex()

```
__gnu_cxx::__mutex & __gnu_debug::_Safe_sequence_base::_M_get_mutex () const  [protected], [noexcept],
[inherited]
```

For use in _Safe_sequence.
Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if(), __gnu_debug::_Safe_unordered_container< _Conta
__gnu_debug::_Safe_unordered_container< _Container >::_M_invalidate_local_if(), and __gnu_debug::_Safe_sequence< _Sequence

_M_invalidate_all()

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all () const  [inline], [protected], [inherited]
```

Invalidates all iterators.
References _M_invalidate_all(), and _M_version.
Referenced by _M_invalidate_all().

_M_invalidate_if()

```
template<typename _Sequence>
template<typename _Predicate>
void __gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if (
            _Predicate __pred) const
```

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal iterators nested in the safe ones.
References __gnu_debug::_Safe_sequence_base::_M_const_iterators, __gnu_debug::_Safe_sequence_base::_M_get_mutex(), and __gnu_debug::_Safe_sequence_base::_M_iterators.

_M_revalidate_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () const  [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

_M_swap()

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
            const _Safe_sequence_base & __x) const  [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

_M_transfer_from_if()

```
template<typename _Sequence>
template<typename _Predicate>
void __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if (
            const _Safe_sequence< _Sequence > & __from,
            _Predicate __pred) const
```

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal iterators nested in the safe ones.

References std::__addressof(), __gnu_debug::_Safe_sequence_base::_M_const_iterators, __gnu_debug::_Safe_sequence_base::_M_ge __gnu_debug::_Safe_sequence_base::_M_iterators, __gnu_debug::_Safe_iterator_base::_M_next, __gnu_debug::_Safe_iterator_base:: __gnu_debug::_Safe_iterator_base::_M_sequence, __gnu_debug::_Safe_iterator_base::_M_version, and __gnu_debug::_Safe_sequence

5.155.3 Member Data Documentation

_M_const_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators  [mutable], [inherited]
```

The list of constant iterators that reference this container.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if(), __gnu_debug::_Safe_unordered_container< _Conta and __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

_M_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators  [mutable], [inherited]
```

The list of mutable iterators that reference this container.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if(), __gnu_debug::_Safe_unordered_container< _Conta and __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

_M_version

```
unsigned int __gnu_debug::_Safe_sequence_base::_M_version  [mutable], [inherited]
```

The container version number. This number may never be 0.

Referenced by _M_invalidate_all(), __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if(), __gnu_debug::_Safe_iterator __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(), __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer > and __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=().

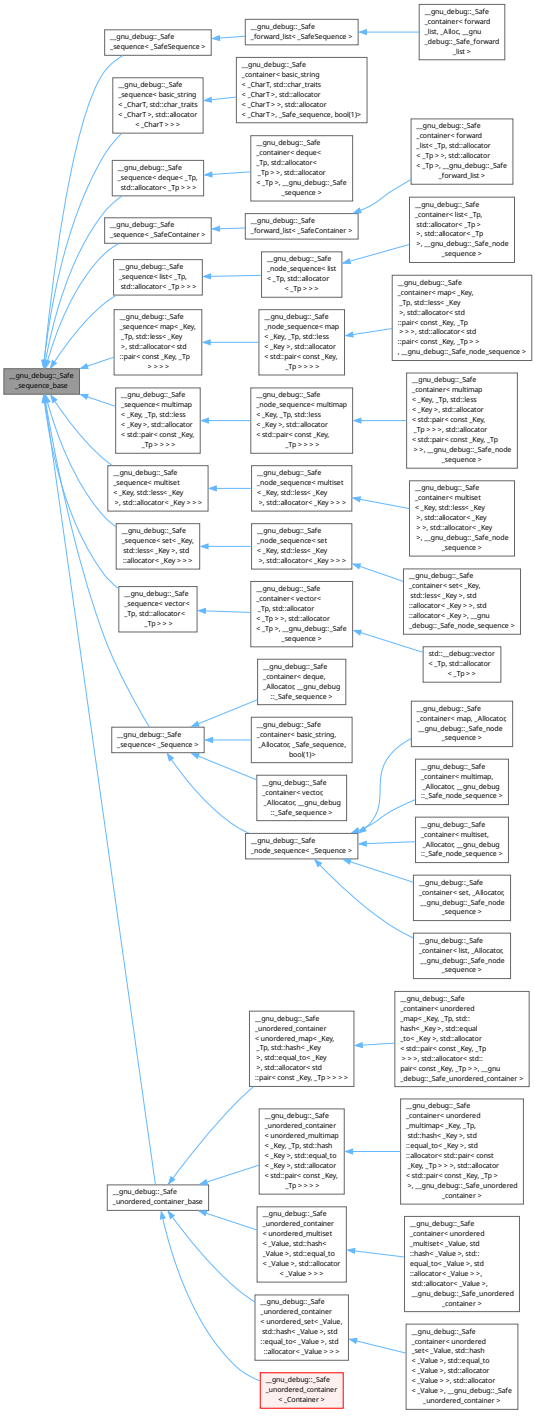
The documentation for this class was generated from the following files:

- formatter.h
- safe_sequence.h
- safe_sequence.tcc

5.156 __gnu_debug::_Safe_sequence_base Class Reference

```
#include <safe_base.h>
```

Inheritance diagram for __gnu_debug::_Safe_sequence_base:


__gnu_debug::_Safe_sequence_base
__gnu_debug::_Safe_sequence< _SafeSequence >
__gnu_debug::_Safe_forward_list< _SafeSequence >
__gnu_debug::_Safe_container< forward_list, _Alloc, __gnu_debug::_Safe_forward_list >
__gnu_debug::_Safe_sequence< basic_string< _CharT, std::char_traits< _CharT >, std::allocator< _CharT > > >
__gnu_debug::_Safe_container< basic_string< _CharT, std::char_traits< _CharT >, std::allocator< _CharT > >, std::allocator< _CharT >, _Safe_sequence, bool(1)>
__gnu_debug::_Safe_sequence< deque< _Tp, std::allocator< _Tp > > >
__gnu_debug::_Safe_container< deque< _Tp, std::allocator< _Tp > >, __gnu_debug::_Safe_sequence >
__gnu_debug::_Safe_container< forward_list< _Tp, std::allocator< _Tp > >, std::allocator< _Tp >, __gnu_debug::_Safe_forward_list >
__gnu_debug::_Safe_sequence< _SafeContainer >
__gnu_debug::_Safe_forward_list< _SafeContainer >
__gnu_debug::_Safe_container< list< _Tp, std::allocator< _Tp > >, std::allocator< _Tp >, __gnu_debug::_Safe_node_sequence >
__gnu_debug::_Safe_sequence< list< _Tp, std::allocator< _Tp > > >
__gnu_debug::_Safe_node_sequence< list< _Tp, std::allocator< _Tp > > >
__gnu_debug::_Safe_sequence< map< _Key, _Tp, std::less< _Key >, std::allocator< std::pair< const _Key, _Tp > > > >
__gnu_debug::_Safe_node_sequence< map< _Key, _Tp, std::less< _Key >, std::allocator< std::pair< const _Key, _Tp > > > >
__gnu_debug::_Safe_container< map< _Key, _Tp, std::less< _Key >, std::allocator< std::pair< const _Key, _Tp > >, std::allocator< std::pair< const _Key, _Tp > >, __gnu_debug::_Safe_node_sequence >
__gnu_debug::_Safe_sequence< multimap< _Key, _Tp, std::less< _Key >, std::allocator< std::pair< const _Key, _Tp > > > >
__gnu_debug::_Safe_node_sequence< multimap< _Key, _Tp, std::less< _Key >, std::allocator< std::pair< const _Key, _Tp > > > >
__gnu_debug::_Safe_container< multimap< _Key, _Tp, std::less< _Key >, std::allocator< std::pair< const _Key, _Tp > > >, std::allocator< std::pair< const _Key, _Tp > >, __gnu_debug::_Safe_node_sequence >
__gnu_debug::_Safe_sequence< multiset< _Key, std::less< _Key >, std::allocator< _Key > > >
__gnu_debug::_Safe_node_sequence< multiset< _Key, std::less< _Key >, std::allocator< _Key > > >
__gnu_debug::_Safe_container< multiset< _Key, std::less< _Key >, std::allocator< _Key > >, std::allocator< _Key >, __gnu_debug::_Safe_node_sequence >
__gnu_debug::_Safe_sequence< set< _Key, std::less< _Key >, std::allocator< _Key > > >
__gnu_debug::_Safe_node_sequence< set< _Key, std::less< _Key >, std::allocator< _Key > > >
__gnu_debug::_Safe_container< set< _Key, std::less< _Key >, std::allocator< _Key > >, std::allocator< _Key >, __gnu_debug::_Safe_node_sequence >
__gnu_debug::_Safe_sequence< vector< _Tp, std::allocator< _Tp > > >
__gnu_debug::_Safe_container< vector< _Tp, std::allocator< _Tp > >, __gnu_debug::_Safe_sequence >
std::__debug::vector< _Tp, std::allocator< _Tp > >
__gnu_debug::_Safe_container< deque, _Allocator, __gnu_debug::_Safe_sequence >
__gnu_debug::_Safe_sequence< _Sequence >
__gnu_debug::_Safe_container< basic_string, _Allocator, _Safe_sequence, bool(1)>
__gnu_debug::_Safe_container< vector, _Allocator, __gnu_debug::_Safe_sequence >
__gnu_debug::_Safe_node_sequence< _Sequence >
__gnu_debug::_Safe_container< map, _Allocator, __gnu_debug::_Safe_node_sequence >
__gnu_debug::_Safe_container< multimap, _Allocator, __gnu_debug::_Safe_node_sequence >
__gnu_debug::_Safe_container< multiset, _Allocator, __gnu_debug::_Safe_node_sequence >
__gnu_debug::_Safe_container< set, _Allocator, __gnu_debug::_Safe_node_sequence >
__gnu_debug::_Safe_container< list, _Allocator, __gnu_debug::_Safe_node_sequence >
__gnu_debug::_Safe_unordered_container_base
__gnu_debug::_Safe_unordered_container< unordered_map< _Key, _Tp, std::hash< _Key >, std::equal_to< _Key >, std::allocator< std::pair< const _Key, _Tp > > > >
__gnu_debug::_Safe_container< unordered_map< _Key, _Tp, std::hash< _Key >, std::equal_to< _Key >, std::allocator< std::pair< const _Key, _Tp > > >, __gnu_debug::_Safe_unordered_container >
__gnu_debug::_Safe_unordered_container< unordered_multimap< _Key, _Tp, std::hash< _Key >, std::equal_to< _Key >, std::allocator< std::pair< const _Key, _Tp > > > >
__gnu_debug::_Safe_container< unordered_multimap< _Key, _Tp, std::hash< _Key >, std::equal_to< _Key >, std::allocator< std::pair< const _Key, _Tp > >, __gnu_debug::_Safe_unordered_container >
__gnu_debug::_Safe_unordered_container< unordered_multiset< _Value, std::hash< _Value >, std::equal_to< _Value >, std::allocator< _Value > > >
__gnu_debug::_Safe_container< unordered_multiset< _Value, std::hash< _Value >, std::equal_to< _Value >, std::allocator< _Value > >, __gnu_debug::_Safe_unordered_container >
__gnu_debug::_Safe_unordered_container< unordered_set< _Value, std::hash< _Value >, std::equal_to< _Value >, std::allocator< _Value > > >
__gnu_debug::_Safe_container< unordered_set< _Value, std::hash< _Value >, std::equal_to< _Value >, std::allocator< _Value > >, std::allocator< _Value > >, __gnu_debug::_Safe_unordered_container >
__gnu_debug::_Safe_unordered_container< _Container >
__gnu_debug::_Safe_container< _Container >
__gnu_debug::_Safe_container< _Container, _Alloc, __gnu_debug::_Safe_node_sequence >
__gnu_debug::_Safe_container< _Container, _Alloc, __gnu_debug::_Safe_sequence >
__gnu_debug::_Safe_container< _Container, _Alloc, __gnu_debug::_Safe_unordered_container >
__gnu_debug::_Safe_container< _Container, _Alloc, __gnu_debug::_Safe_node_sequence, _Safe_sequence >
__gnu_debug::_Safe_container< _Container, _Alloc, __gnu_debug::_Safe_node_sequence, _Safe_unordered_container >
__gnu_debug::_Safe_container< _Container, _Alloc, __gnu_debug::_Safe_sequence, _Safe_unordered_container >
__gnu_debug::_Safe_container< _Container, _Alloc, __gnu_debug::_Safe_node_sequence, _Safe_sequence, _Safe_unordered_container >
__gnu_debug::_Safe_container< _Container, _Alloc, __gnu_debug::_Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence >
__gnu_debug::_Safe_container< _Container, _Alloc, __gnu_debug::_Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence >
__gnu_debug::_Safe_container< _Container, _Alloc, __gnu_debug::_Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence, _Safe_unordered_container >
__gnu_debug::_Safe_container< _Container, _Alloc, __gnu_debug::_Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence >
__gnu_debug::_Safe_container< _Container, _Alloc, __gnu_debug::_Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence >
__gnu_debug::_Safe_container< _Container, _Alloc, __gnu_debug::_Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence, _Safe_unordered_container >
__gnu_debug::_Safe_container< _Container, _Alloc, __gnu_debug::_Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence >
__gnu_debug::_Safe_container< _Container, _Alloc, __gnu_debug::_Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence >
__gnu_debug::_Safe_container< _Container, _Alloc, __gnu_debug::_Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence, _Safe_unordered_container >
__gnu_debug::_Safe_container< _Container, _Alloc, __gnu_debug::_Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence >
__gnu_debug::_Safe_container< _Container, _Alloc, __gnu_debug::_Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence >
__gnu_debug::_Safe_container< _Container, _Alloc, __gnu_debug::_Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence, _Safe_unordered_container >
__gnu_debug::_Safe_container< _Container, _Alloc, __gnu_debug::_Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence >
__gnu_debug::_Safe_container< _Container, _Alloc, __gnu_debug::_Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence >
__gnu_debug::_Safe_container< _Container, _Alloc, __gnu_debug::_Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence, _Safe_unordered_container >
__gnu_debug::_Safe_container< _Container, _Alloc, __gnu_debug::_Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _Safe_node_sequence >
__gnu_debug::_Safe_container< _Container, _Alloc, __gnu_debug::_Safe_node_sequence, _Safe_sequence, _Safe_unordered_container, _

Protected Member Functions

- constexpr **_Safe_sequence_base** (_Safe_sequence_base &&__seq) noexcept
- constexpr **_Safe_sequence_base** (const _Safe_sequence_base &) noexcept
- constexpr ~_Safe_sequence_base ()
- void _M_detach_all () const
- void _M_detach_singular () const
- __gnu_cxx::__mutex & _M_get_mutex () const noexcept
- void _M_invalidate_all () const
- void _M_revalidate_singular () const
- void _M_swap (const _Safe_sequence_base &__x) const noexcept

Friends

- class **_Safe_iterator_base**

5.156.1 Detailed Description

Base class that supports tracking of iterators that reference a sequence.
The _Safe_sequence_base class provides basic support for tracking iterators into a sequence. Sequences that track iterators must derived from _Safe_sequence_base publicly, so that safe iterators (which inherit _Safe_iterator_base) can attach to them. This class contains two linked lists of iterators, one for constant iterators and one for mutable iterators, and a version number that allows very fast invalidation of all iterators that reference the container.
This class must ensure that no operation on it may throw an exception, otherwise *safe* sequences may fail to provide the exception-safety guarantees required by the C++ standard.

5.156.2 Constructor & Destructor Documentation

~_Safe_sequence_base()

```
__gnu_debug::_Safe_sequence_base::~_Safe_sequence_base ()  [inline], [constexpr], [protected]
```

Notify all iterators that reference this sequence that the sequence is being destroyed.
References _M_detach_all().

5.156.3 Member Function Documentation

_M_detach_all()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_all () const  [protected]
```

Detach all iterators, leaving them singular.
Referenced by ~_Safe_sequence_base().

_M_detach_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular () const  [protected]
```

Detach all singular iterators.

Postcondition

for all iterators i attached to this sequence, i->_M_version == _M_version.

_M_get_mutex()

```
__gnu_cxx::__mutex & __gnu_debug::_Safe_sequence_base::_M_get_mutex () const  [protected], [noexcept]
```

For use in _Safe_sequence.
Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if(), __gnu_debug::_Safe_unordered_container< _Conta
__gnu_debug::_Safe_unordered_container< _Container >::_M_invalidate_local_if(), and __gnu_debug::_Safe_sequence< _Sequence

_M_invalidate_all()

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all () const  [inline], [protected]
```

Invalidates all iterators.
References _M_invalidate_all(), and _M_version.
Referenced by _M_invalidate_all().

_M_revalidate_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () const  [protected]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

_M_swap()

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
             const _Safe_sequence_base & __x) const  [protected], [noexcept]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

5.156.4 Member Data Documentation

_M_const_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators  [mutable]
```

The list of constant iterators that reference this container.
Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if(), __gnu_debug::_Safe_unordered_container< _Conta
and __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

_M_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators  [mutable]
```

The list of mutable iterators that reference this container.
Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if(), __gnu_debug::_Safe_unordered_container< _Conta
and __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

_M_version

```
unsigned int __gnu_debug::_Safe_sequence_base::_M_version  [mutable]
```

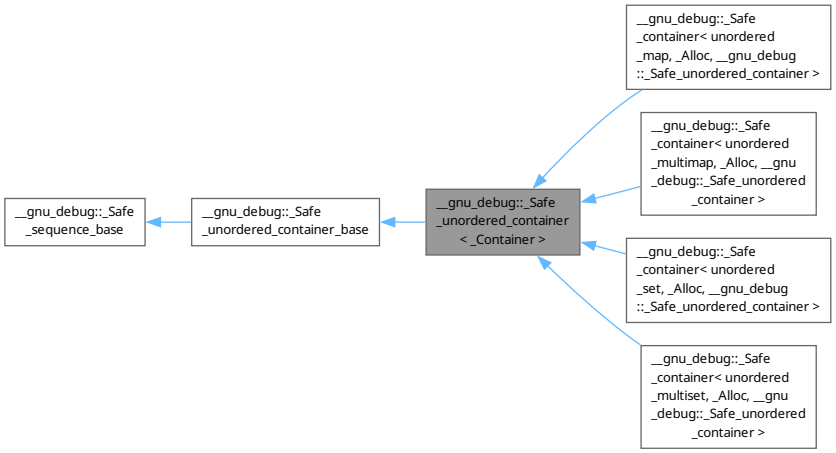
The container version number. This number may never be 0.
Referenced by _M_invalidate_all(), __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if(), __gnu_debug::_Safe_iterator
__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(), __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >
and __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=().
The documentation for this class was generated from the following file:

- safe_base.h

5.157 __gnu_debug::_Safe_unordered_container< _Container > Class Template Reference

```
#include <safe_unordered_container.h>
```

Inheritance diagram for __gnu_debug::_Safe_unordered_container< _Container >:



Public Member Functions

- void **_M_invalidate_all** ()
- template<typename _Predicate>
  void _M_invalidate_if (_Predicate __pred)
- template<typename _Predicate>
  void _M_invalidate_local_if (_Predicate __pred)

Public Attributes

- _Safe_iterator_base * _M_const_iterators
- _Safe_iterator_base * _M_const_local_iterators
- _Safe_iterator_base * _M_iterators
- _Safe_iterator_base * _M_local_iterators
- unsigned int _M_version

Protected Member Functions

- void _M_detach_all () const
- void _M_detach_singular () const
- __gnu_cxx::__mutex & _M_get_mutex () const noexcept
- void _M_invalidate_all () const
- void **_M_invalidate_locals** ()
- void _M_revalidate_singular () const
- void _M_swap (const _Safe_sequence_base &__x) const noexcept
- void _M_swap (const _Safe_unordered_container_base &__x) const noexcept

Static Protected Member Functions

- template<typename _ExtractKey, typename _Source>
  static _UContMergeGuard< _Source, _UContInvalidatePred< _ExtractKey, _Source > > **_S_uc_guard** (_ExtractKey, _Source &__src)

- template<typename _ExtractKey, typename _Source>
  static _UContMergeGuard< _Source, _UMContInvalidatePred< _ExtractKey, _Source > > **_S_umc_guard** (↩
  _ExtractKey, _Source &__src)

5.157.1 Detailed Description

template<typename _Container>
class __gnu_debug::_Safe_unordered_container< _Container >

Base class for constructing a *safe* unordered container type that tracks iterators that reference it.
The class template _Safe_unordered_container simplifies the construction of *safe* unordered containers that track the iterators that reference the container, so that the iterators are notified of changes in the container that may affect their operation, e.g., if the container invalidates its iterators or is destructed. This class template may only be used by deriving from it and passing the name of the derived class as its template parameter via the curiously recurring template pattern. The derived class must have `iterator` and `const_iterator` types that are instantiations of class template _↩
Safe_iterator for this container and `local_iterator` and `const_local_iterator` types that are instantiations of class template _Safe_local_iterator for this container. Iterators will then be tracked automatically.

5.157.2 Member Function Documentation

_M_detach_all()

```
void __gnu_debug::_Safe_unordered_container_base::_M_detach_all () const  [protected], [inherited]
```

Detach all iterators, leaving them singular.
Referenced by ~_Safe_unordered_container_base().

_M_detach_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular () const  [protected], [inherited]
```

Detach all singular iterators.

Postcondition

for all iterators i attached to this sequence, i->_M_version == _M_version.

_M_get_mutex()

```
__gnu_cxx::__mutex & __gnu_debug::_Safe_sequence_base::_M_get_mutex () const  [protected], [noexcept],
[inherited]
```

For use in _Safe_sequence.
Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if(), __gnu_debug::_Safe_unordered_container< _Conta
__gnu_debug::_Safe_unordered_container< _Container >::_M_invalidate_local_if(), and __gnu_debug::_Safe_sequence< _Sequence

_M_invalidate_all()

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all () const  [inline], [protected], [inherited]
```

Invalidates all iterators.
References _M_invalidate_all(), and _M_version.
Referenced by _M_invalidate_all().

_M_invalidate_if()

```
template<typename _Container>
template<typename _Predicate>
void __gnu_debug::_Safe_unordered_container< _Container >::_M_invalidate_if (
            _Predicate __pred)
```

Invalidates all iterators `x` that reference this container, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal iterators nested in the safe ones.
References __gnu_debug::_Safe_sequence_base::_M_const_iterators, __gnu_debug::_Safe_sequence_base::_M_get_mutex(), and __gnu_debug::_Safe_sequence_base::_M_iterators.

_M_invalidate_local_if()

```
template<typename _Container>
template<typename _Predicate>
void __gnu_debug::_Safe_unordered_container< _Container >::_M_invalidate_local_if (
            _Predicate __pred)
```

Invalidates all local iterators `x` that reference this container, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal local iterators nested in the safe ones.
References __gnu_debug::_Safe_unordered_container_base::_M_const_local_iterators, __gnu_debug::_Safe_sequence_base::_M_get_
and __gnu_debug::_Safe_unordered_container_base::_M_local_iterators.

_M_revalidate_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () const  [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

_M_swap() [1/2]

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
            const _Safe_sequence_base & __x) const  [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

_M_swap() [2/2]

```
void __gnu_debug::_Safe_unordered_container_base::_M_swap (
            const _Safe_unordered_container_base & __x) const  [protected], [noexcept], [inherited]
```

Swap this container with the given container. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

5.157.3 Member Data Documentation

_M_const_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators  [mutable], [inherited]
```

The list of constant iterators that reference this container.
Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if(), __gnu_debug::_Safe_unordered_container< _Conta
and __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

_M_const_local_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_const_local_iterators  [mutable],
[inherited]
```

The list of constant local iterators that reference this container.
Referenced by __gnu_debug::_Safe_unordered_container< _Container >::_M_invalidate_local_if().

_M_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators  [mutable], [inherited]
```

The list of mutable iterators that reference this container.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if(), __gnu_debug::_Safe_unordered_container< _Conta
and __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

_M_local_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_local_iterators  [mutable],
[inherited]
```

The list of mutable local iterators that reference this container.
Referenced by __gnu_debug::_Safe_unordered_container< _Container >::_M_invalidate_local_if().

_M_version

```
unsigned int __gnu_debug::_Safe_sequence_base::_M_version  [mutable], [inherited]
```

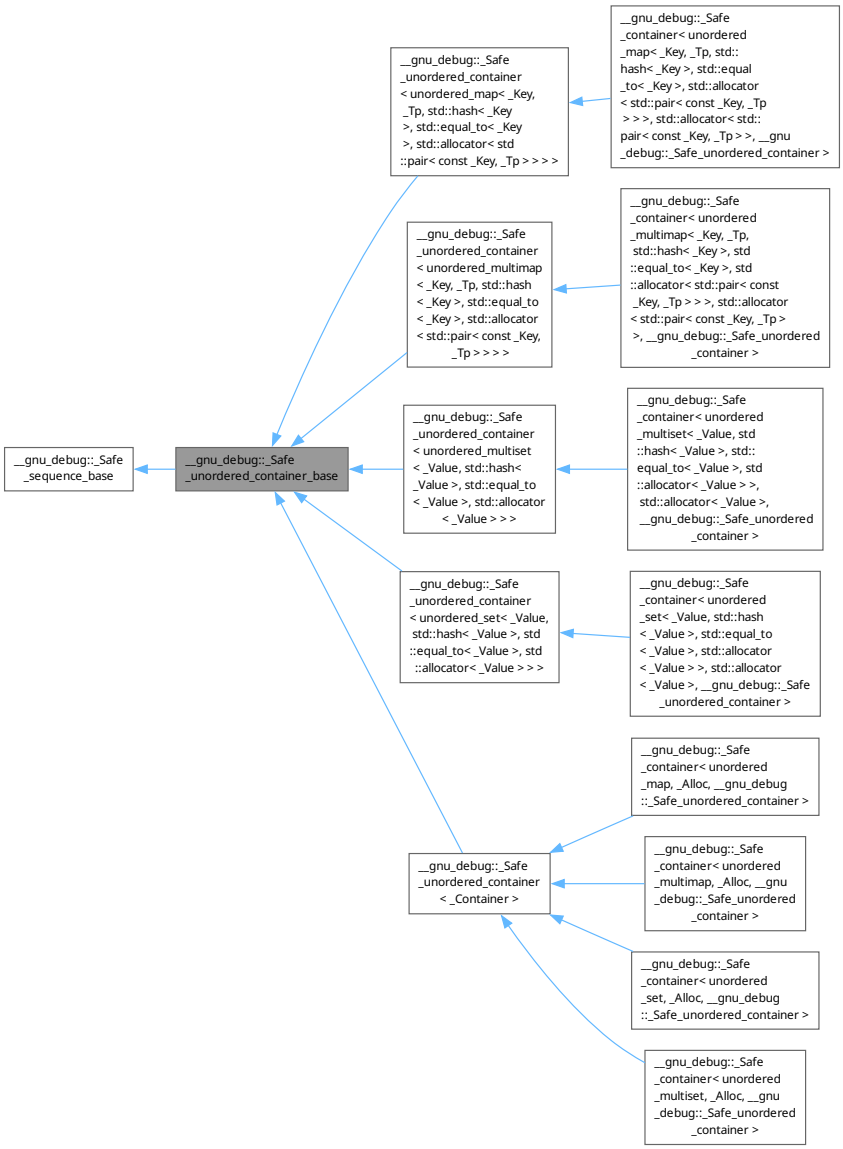
The container version number. This number may never be 0.
Referenced by _M_invalidate_all(), __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if(), __gnu_debug::_Safe_iterator
__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(), __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >
and __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=().
The documentation for this class was generated from the following files:

- safe_unordered_container.h
- safe_unordered_container.tcc

5.158 __gnu_debug::_Safe_unordered_container_base Class Reference

```
#include <safe_unordered_base.h>
```

Inheritance diagram for __gnu_debug::_Safe_unordered_container_base:



Public Attributes

- _Safe_iterator_base * _M_const_iterators
- _Safe_iterator_base * _M_const_local_iterators
- _Safe_iterator_base * _M_iterators
- _Safe_iterator_base * _M_local_iterators
- unsigned int _M_version

Protected Member Functions

- **_Safe_unordered_container_base** (_Safe_unordered_container_base &&__x) noexcept

- **_Safe_unordered_container_base** (const _Safe_unordered_container_base &) noexcept
- ~_Safe_unordered_container_base () noexcept
- void _M_detach_all () const
- void _M_detach_singular () const
- __gnu_cxx::__mutex & _M_get_mutex () const noexcept
- void _M_invalidate_all () const
- void _M_revalidate_singular () const
- void _M_swap (const _Safe_sequence_base &__x) const noexcept
- void _M_swap (const _Safe_unordered_container_base &__x) const noexcept

Friends

- class **_Safe_local_iterator_base**

5.158.1 Detailed Description

Base class that supports tracking of local iterators that reference an unordered container.
The _Safe_unordered_container_base class provides basic support for tracking iterators into an unordered container. Containers that track iterators must derived from _Safe_unordered_container_base publicly, so that safe iterators (which inherit _Safe_iterator_base) can attach to them. This class contains four linked lists of iterators, one for constant iterators, one for mutable iterators, one for constant local iterators, one for mutable local iterators and a version number that allows very fast invalidation of all iterators that reference the container.
This class must ensure that no operation on it may throw an exception, otherwise *safe* containers may fail to provide the exception-safety guarantees required by the C++ standard.

5.158.2 Constructor & Destructor Documentation

~_Safe_unordered_container_base()

```
__gnu_debug::_Safe_unordered_container_base::~_Safe_unordered_container_base ()  [inline], [protected],
[noexcept]
```

Notify all iterators that reference this container that the container is being destroyed.
References _M_detach_all().

5.158.3 Member Function Documentation

_M_detach_all()

```
void __gnu_debug::_Safe_unordered_container_base::_M_detach_all () const  [protected]
```

Detach all iterators, leaving them singular.
Referenced by ~_Safe_unordered_container_base().

_M_detach_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular () const  [protected], [inherited]
```

Detach all singular iterators.

Postcondition

for all iterators i attached to this sequence, i->_M_version == _M_version.

_M_get_mutex()

```
__gnu_cxx::__mutex & __gnu_debug::_Safe_sequence_base::_M_get_mutex () const  [protected], [noexcept],
[inherited]
```

For use in _Safe_sequence.
Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if(), __gnu_debug::_Safe_unordered_container< _Conta
__gnu_debug::_Safe_unordered_container< _Container >::_M_invalidate_local_if(), and __gnu_debug::_Safe_sequence< _Sequence

_M_invalidate_all()

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all () const  [inline], [protected], [inherited]
```

Invalidates all iterators.
References _M_invalidate_all(), and _M_version.
Referenced by _M_invalidate_all().

_M_revalidate_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () const  [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

_M_swap() [1/2]

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
            const _Safe_sequence_base & __x) const  [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

_M_swap() [2/2]

```
void __gnu_debug::_Safe_unordered_container_base::_M_swap (
            const _Safe_unordered_container_base & __x) const  [protected], [noexcept]
```

Swap this container with the given container. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

5.158.4 Member Data Documentation

_M_const_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators  [mutable], [inherited]
```

The list of constant iterators that reference this container.
Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if(), __gnu_debug::_Safe_unordered_container< _Conta
and __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

_M_const_local_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_const_local_iterators  [mutable]
```

The list of constant local iterators that reference this container.
Referenced by __gnu_debug::_Safe_unordered_container< _Container >::_M_invalidate_local_if().

_M_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators  [mutable], [inherited]
```

The list of mutable iterators that reference this container.
Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if(), __gnu_debug::_Safe_unordered_container< _Conta
and __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

_M_local_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_local_iterators  [mutable]
```

The list of mutable local iterators that reference this container.
Referenced by __gnu_debug::_Safe_unordered_container< _Container >::_M_invalidate_local_if().

_M_version

```
unsigned int __gnu_debug::_Safe_sequence_base::_M_version  [mutable], [inherited]
```

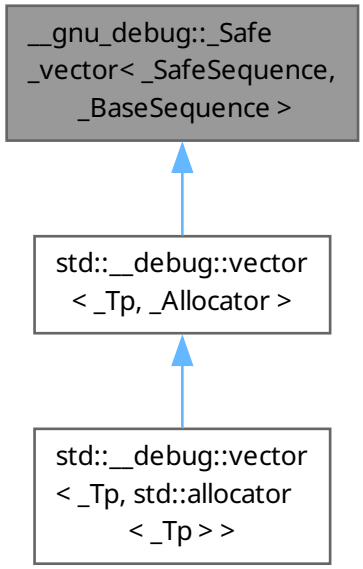
The container version number. This number may never be 0.
Referenced by _M_invalidate_all(), __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if(), __gnu_debug::_Safe_iterator
__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(), __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >
and __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=().
The documentation for this class was generated from the following file:

- safe_unordered_base.h

5.159 __gnu_debug::_Safe_vector< _SafeSequence, _BaseSequence > Class Template Reference

```
#include <vector>
```

Inheritance diagram for __gnu_debug::_Safe_vector< _SafeSequence, _BaseSequence >:



Protected Member Functions

- constexpr **_Safe_vector** (_Safe_vector &&__x) noexcept
- constexpr **_Safe_vector** (const _Safe_vector &) noexcept
- constexpr **_Safe_vector** (size_type __n) noexcept
- bool **_M_requires_reallocation** (size_type __elements) const noexcept

- constexpr void **_M_update_guaranteed_capacity** () noexcept
- constexpr _Safe_vector & **operator=** (_Safe_vector &&__x) noexcept
- constexpr _Safe_vector & **operator=** (const _Safe_vector &) noexcept

Protected Attributes

- size_type **_M_guaranteed_capacity**

5.159.1 Detailed Description

template<typename _SafeSequence, typename _BaseSequence>
class __gnu_debug::_Safe_vector< _SafeSequence, _BaseSequence >

Base class for Debug Mode vector.
Adds information about the guaranteed capacity, which is useful for detecting code which relies on non-portable implementation details of the libstdc++ reallocation policy.
The documentation for this class was generated from the following file:

- debug/vector

5.160 __gnu_parallel::_SamplingSorter< __stable, _RAIter, _StrictWeakOrdering > Struct Template Reference

`#include <multiway_merge.h>`

Public Member Functions

- void **operator()** (_RAIter __first, _RAIter __last, _StrictWeakOrdering __comp)

5.160.1 Detailed Description

template<bool __stable, class _RAIter, class _StrictWeakOrdering>
struct __gnu_parallel::_SamplingSorter< __stable, _RAIter, _StrictWeakOrdering >

Stable sorting functor.
Used to reduce code instanciation in multiway_merge_sampling_splitting.
The documentation for this struct was generated from the following file:

- multiway_merge.h

5.161 __gnu_parallel::_SamplingSorter< false, _RAIter, _StrictWeakOrdering > Struct Template Reference

`#include <multiway_merge.h>`

Public Member Functions

- void **operator()** (_RAIter __first, _RAIter __last, _StrictWeakOrdering __comp)
- void **operator()** (_RAIter __first, _RAIter __last, _StrictWeakOrdering __comp)

5.161.1 Detailed Description

template<class _RAIter, class _StrictWeakOrdering>
struct __gnu_parallel::_SamplingSorter< false, _RAIter, _StrictWeakOrdering >

Non-__stable sorting functor.

Used to reduce code instantiation in multiway_merge_sampling_splitting.
The documentation for this struct was generated from the following file:

- multiway_merge.h

5.162 std::__detail::_Scanner< _CharT > Class Template Reference

`#include <regex_scanner.h>`

Public Types

- typedef const std::ctype< _CharT > **_CtypeT**
- typedef regex_constants::syntax_option_type **_FlagT**
- typedef std::basic_string< _CharT > **_StringT**
- enum **_TokenT** : unsigned {
  _S_token_anychar , **_S_token_ord_char** , **_S_token_oct_num** , **_S_token_hex_num** ,
  _S_token_backref , **_S_token_subexpr_begin** , **_S_token_subexpr_no_group_begin** , **_S_token_↩ subexpr_lookahead_begin** ,
  _S_token_subexpr_end , **_S_token_bracket_begin** , **_S_token_bracket_neg_begin** , **_S_token_bracket_↩ end** ,
  _S_token_interval_begin , **_S_token_interval_end** , **_S_token_quoted_class** , **_S_token_char_class_name** ,
  _S_token_collsymbol , **_S_token_equiv_class_name** , **_S_token_opt** , **_S_token_or** ,
  _S_token_closure0 , **_S_token_closure1** , **_S_token_line_begin** , **_S_token_line_end** ,
  _S_token_word_bound , **_S_token_comma** , **_S_token_dup_count** , **_S_token_eof** ,
  _S_token_bracket_dash , **_S_token_unknown** }

Public Member Functions

- **_Scanner** (const _CharT *__begin, const _CharT *__end, _FlagT __flags, std::locale __loc)
- void **_M_advance** ()
- _TokenT **_M_get_token** () const noexcept
- const _StringT & **_M_get_value** () const noexcept

Protected Types

- enum **_StateT** { **_S_state_normal** , **_S_state_in_brace** , **_S_state_in_bracket** }

Protected Member Functions

- const char * **_M_find_escape** (char __c)
- bool **_M_is_awk** () const
- bool **_M_is_basic** () const
- bool **_M_is_ecma** () const
- bool **_M_is_extended** () const
- bool **_M_is_grep** () const

Protected Attributes

- bool **_M_at_bracket_start**
- const std::pair< char, char > **_M_awk_escape_tbl** [11]
- const char * **_M_basic_spec_char**
- const std::pair< char, char > **_M_ecma_escape_tbl** [8]
- const char * **_M_ecma_spec_char**

- const std::pair< char, char > * **_M_escape_tbl**
- const char * **_M_extended_spec_char**
- _FlagT **_M_flags**
- const char * **_M_spec_char**
- _StateT **_M_state**
- _TokenT **_M_token**
- const std::pair< char, _TokenT > **_M_token_tbl** [9]

5.162.1 Detailed Description

template<typename _CharT>
class std::__detail::_Scanner< _CharT >

Scans an input range for regex tokens.
The _Scanner class interprets the regular expression pattern in the input range passed to its constructor as a sequence of parse tokens passed to the regular expression compiler. The sequence of tokens provided depends on the flag settings passed to the constructor: different regular expression grammars will interpret the same input pattern in syntactically different ways.

5.162.2 Member Enumeration Documentation

_TokenT

```
enum std::__detail::_ScannerBase::_TokenT : unsigned  [inherited]
```

Token types returned from the scanner.
The documentation for this class was generated from the following files:

- regex_scanner.h
- regex_scanner.tcc

5.163 __gnu_debug::_Sequence_traits< _Sequence > Struct Template Reference

```
#include <safe_iterator.h>
```

Public Types

- typedef _Distance_traits< typename _Sequence::iterator > **_DistTraits**

Static Public Member Functions

- static _DistTraits::__type **_S_size** (const _Sequence &__seq)

5.163.1 Detailed Description

template<typename _Sequence>
struct __gnu_debug::_Sequence_traits< _Sequence >

Sequence traits giving the size of a container if possible.
The documentation for this struct was generated from the following file:

- safe_iterator.h

5.164 __gnu_parallel::_Settings Struct Reference

```
#include <settings.h>
```

Static Public Member Functions

- static const _Settings & get () throw ()
- static void set (_Settings &) throw ()

Public Attributes

- _SequenceIndex accumulate_minimal_n
- unsigned int adjacent_difference_minimal_n
- _AlgorithmStrategy **algorithm_strategy**
- unsigned int cache_line_size
- _SequenceIndex count_minimal_n
- _SequenceIndex fill_minimal_n
- _FindAlgorithm **find_algorithm**
- double find_increasing_factor
- _SequenceIndex find_initial_block_size
- _SequenceIndex find_maximum_block_size
- float find_scale_factor
- _SequenceIndex find_sequential_search_size
- _SequenceIndex for_each_minimal_n
- _SequenceIndex generate_minimal_n
- unsigned long long L1_cache_size
- unsigned long long L2_cache_size
- _SequenceIndex max_element_minimal_n
- _SequenceIndex merge_minimal_n
- unsigned int merge_oversampling
- _SplittingAlgorithm **merge_splitting**
- _SequenceIndex min_element_minimal_n
- _MultiwayMergeAlgorithm **multiway_merge_algorithm**
- int multiway_merge_minimal_k
- _SequenceIndex multiway_merge_minimal_n
- unsigned int multiway_merge_oversampling
- _SplittingAlgorithm **multiway_merge_splitting**
- _SequenceIndex nth_element_minimal_n
- _SequenceIndex partial_sort_minimal_n
- _PartialSumAlgorithm **partial_sum_algorithm**
- float partial_sum_dilation
- unsigned int partial_sum_minimal_n
- double partition_chunk_share
- _SequenceIndex partition_chunk_size
- _SequenceIndex partition_minimal_n
- _SequenceIndex qsb_steals
- unsigned int random_shuffle_minimal_n
- _SequenceIndex replace_minimal_n
- _SequenceIndex search_minimal_n
- _SequenceIndex set_difference_minimal_n
- _SequenceIndex set_intersection_minimal_n
- _SequenceIndex set_symmetric_difference_minimal_n
- _SequenceIndex set_union_minimal_n
- _SortAlgorithm **sort_algorithm**
- _SequenceIndex sort_minimal_n
- unsigned int sort_mwms_oversampling

- unsigned int sort_qs_num_samples_preset
- _SequenceIndex sort_qsb_base_case_maximal_n
- _SplittingAlgorithm **sort_splitting**
- unsigned int TLB_size
- _SequenceIndex transform_minimal_n
- _SequenceIndex unique_copy_minimal_n
- _SequenceIndex **workstealing_chunk_size**

5.164.1 Detailed Description

class _Settings Run-time settings for the parallel mode including all tunable parameters.

5.164.2 Member Function Documentation

get()

```
static const _Settings & __gnu_parallel::_Settings::get () throw ( )   [static]
```

Get the global settings.
Referenced by __gnu_parallel::__find_template(), __gnu_parallel::__find_template(), __gnu_parallel::__find_template(), __gnu_parallel::__for_each_template_random_access_workstealing(), __gnu_parallel::__parallel_nth_element(), __gnu_parallel::__parallel_partial_sum(), __gnu_parallel::__parallel_partial_sum_linear(), __gnu_parallel::__parallel_partition(), __gnu_parallel::__parallel_random_shuffle_drs(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort_qs_conquer(), __gnu_parallel::__qsb_local_sort_with_helping(), __gnu_parallel::__sequential_random_s __gnu_parallel::multiway_merge_sampling_splitting(), __gnu_parallel::parallel_multiway_merge(), __gnu_parallel::parallel_sort_mwms(), and __gnu_parallel::parallel_sort_mwms_pu().

set()

```
static void __gnu_parallel::_Settings::set (
            _Settings & ) throw ( )   [static]
```

Set the global settings.
References accumulate_minimal_n, adjacent_difference_minimal_n, cache_line_size, count_minimal_n, fill_minimal_n, find_increasing_factor, find_initial_block_size, find_maximum_block_size, find_scale_factor, find_sequential_search_size, for_each_minimal_n, generate_minimal_n, L1_cache_size, L2_cache_size, max_element_minimal_n, merge_minimal_n, merge_oversampling, min_element_minimal_n, multiway_merge_minimal_k, multiway_merge_minimal_n, multiway_merge_oversampling nth_element_minimal_n, partial_sort_minimal_n, partial_sum_dilation, partial_sum_minimal_n, partition_chunk_share, partition_chunk_size, partition_minimal_n, qsb_steals, random_shuffle_minimal_n, replace_minimal_n, search_minimal_n, set(), set_difference_minimal_n, set_intersection_minimal_n, set_symmetric_difference_minimal_n, set_union_minimal_n, sort_minimal_n, sort_mwms_oversampling, sort_qs_num_samples_preset, sort_qsb_base_case_maximal_n, TLB_size, transform_minimal_n, and unique_copy_minimal_n.
Referenced by set().

5.164.3 Member Data Documentation

accumulate_minimal_n

```
_SequenceIndex __gnu_parallel::_Settings::accumulate_minimal_n
```

Minimal input size for accumulate.
Referenced by set().

adjacent_difference_minimal_n

```
unsigned int __gnu_parallel::_Settings::adjacent_difference_minimal_n
```

Minimal input size for adjacent_difference.
Referenced by set().

cache_line_size

```
unsigned int __gnu_parallel::_Settings::cache_line_size
```
Overestimation of cache line size. Used to avoid false sharing, i.e. elements of different threads are at least this amount apart.
Referenced by __gnu_parallel::__for_each_template_random_access_workstealing(), and set().

count_minimal_n

```
_SequenceIndex __gnu_parallel::_Settings::count_minimal_n
```
Minimal input size for count and count_if.
Referenced by set().

fill_minimal_n

```
_SequenceIndex __gnu_parallel::_Settings::fill_minimal_n
```
Minimal input size for fill.
Referenced by set().

find_increasing_factor

```
double __gnu_parallel::_Settings::find_increasing_factor
```
Block size increase factor for find.
Referenced by set().

find_initial_block_size

```
_SequenceIndex __gnu_parallel::_Settings::find_initial_block_size
```
Initial block size for find.
Referenced by __gnu_parallel::__find_template(), and set().

find_maximum_block_size

```
_SequenceIndex __gnu_parallel::_Settings::find_maximum_block_size
```
Maximal block size for find.
Referenced by set().

find_scale_factor

```
float __gnu_parallel::_Settings::find_scale_factor
```
Block size scale-down factor with respect to current position.
Referenced by __gnu_parallel::__find_template(), and set().

find_sequential_search_size

```
_SequenceIndex __gnu_parallel::_Settings::find_sequential_search_size
```
Start with looking for this many elements sequentially, for find.
Referenced by __gnu_parallel::__find_template(), __gnu_parallel::__find_template(), and set().

for_each_minimal_n

```
_SequenceIndex __gnu_parallel::_Settings::for_each_minimal_n
```
Minimal input size for for_each.
Referenced by set().

generate_minimal_n

_SequenceIndex __gnu_parallel::_Settings::generate_minimal_n
Minimal input size for generate.
Referenced by set().

L1_cache_size

unsigned long long __gnu_parallel::_Settings::L1_cache_size
size of the L1 cache in bytes (underestimation).
Referenced by set().

L2_cache_size

unsigned long long __gnu_parallel::_Settings::L2_cache_size
size of the L2 cache in bytes (underestimation).
Referenced by __gnu_parallel::__parallel_random_shuffle_drs(), __gnu_parallel::__sequential_random_shuffle(), and set().

max_element_minimal_n

_SequenceIndex __gnu_parallel::_Settings::max_element_minimal_n
Minimal input size for max_element.
Referenced by set().

merge_minimal_n

_SequenceIndex __gnu_parallel::_Settings::merge_minimal_n
Minimal input size for merge.
Referenced by set().

merge_oversampling

unsigned int __gnu_parallel::_Settings::merge_oversampling
Oversampling factor for merge.
Referenced by __gnu_parallel::multiway_merge_sampling_splitting(), __gnu_parallel::parallel_multiway_merge(), and set().

min_element_minimal_n

_SequenceIndex __gnu_parallel::_Settings::min_element_minimal_n
Minimal input size for min_element.
Referenced by set().

multiway_merge_minimal_k

int __gnu_parallel::_Settings::multiway_merge_minimal_k
Oversampling factor for multiway_merge.
Referenced by set().

multiway_merge_minimal_n

_SequenceIndex __gnu_parallel::_Settings::multiway_merge_minimal_n
Minimal input size for multiway_merge.
Referenced by set().

multiway_merge_oversampling

`unsigned int __gnu_parallel::_Settings::multiway_merge_oversampling`
Oversampling factor for multiway_merge.
Referenced by set().

nth_element_minimal_n

`_SequenceIndex __gnu_parallel::_Settings::nth_element_minimal_n`
Minimal input size for nth_element.
Referenced by __gnu_parallel::__parallel_nth_element(), and set().

partial_sort_minimal_n

`_SequenceIndex __gnu_parallel::_Settings::partial_sort_minimal_n`
Minimal input size for partial_sort.
Referenced by set().

partial_sum_dilation

`float __gnu_parallel::_Settings::partial_sum_dilation`
Ratio for partial_sum. Assume "sum and write result" to be this factor slower than just "sum".
Referenced by __gnu_parallel::__parallel_partial_sum_linear(), and set().

partial_sum_minimal_n

`unsigned int __gnu_parallel::_Settings::partial_sum_minimal_n`
Minimal input size for partial_sum.
Referenced by set().

partition_chunk_share

`double __gnu_parallel::_Settings::partition_chunk_share`
Chunk size for partition, relative to input size. If > 0.0, this value overrides partition_chunk_size.
Referenced by __gnu_parallel::__parallel_partition(), and set().

partition_chunk_size

`_SequenceIndex __gnu_parallel::_Settings::partition_chunk_size`
Chunk size for partition.
Referenced by __gnu_parallel::__parallel_partition(), and set().

partition_minimal_n

`_SequenceIndex __gnu_parallel::_Settings::partition_minimal_n`
Minimal input size for partition.
Referenced by __gnu_parallel::__parallel_nth_element(), and set().

qsb_steals

`_SequenceIndex __gnu_parallel::_Settings::qsb_steals`
The number of stolen ranges in load-balanced quicksort.
Referenced by set().

random_shuffle_minimal_n

`unsigned int __gnu_parallel::_Settings::random_shuffle_minimal_n`
Minimal input size for random_shuffle.
Referenced by set().

replace_minimal_n

`_SequenceIndex __gnu_parallel::_Settings::replace_minimal_n`
Minimal input size for replace and replace_if.
Referenced by set().

search_minimal_n

`_SequenceIndex __gnu_parallel::_Settings::search_minimal_n`
Minimal input size for search and search_n.
Referenced by set().

set_difference_minimal_n

`_SequenceIndex __gnu_parallel::_Settings::set_difference_minimal_n`
Minimal input size for set_difference.
Referenced by set().

set_intersection_minimal_n

`_SequenceIndex __gnu_parallel::_Settings::set_intersection_minimal_n`
Minimal input size for set_intersection.
Referenced by set().

set_symmetric_difference_minimal_n

`_SequenceIndex __gnu_parallel::_Settings::set_symmetric_difference_minimal_n`
Minimal input size for set_symmetric_difference.
Referenced by set().

set_union_minimal_n

`_SequenceIndex __gnu_parallel::_Settings::set_union_minimal_n`
Minimal input size for set_union.
Referenced by set().

sort_minimal_n

`_SequenceIndex __gnu_parallel::_Settings::sort_minimal_n`
Minimal input size for parallel sorting.
Referenced by set().

sort_mwms_oversampling

`unsigned int __gnu_parallel::_Settings::sort_mwms_oversampling`
Oversampling factor for parallel std::sort (MWMS).
Referenced by __gnu_parallel::parallel_sort_mwms(), __gnu_parallel::parallel_sort_mwms_pu(), and set().

sort_qs_num_samples_preset

unsigned int __gnu_parallel::_Settings::sort_qs_num_samples_preset
Such many samples to take to find a good pivot (quicksort).
Referenced by set().

sort_qsb_base_case_maximal_n

_SequenceIndex __gnu_parallel::_Settings::sort_qsb_base_case_maximal_n
Maximal subsequence __length to switch to unbalanced __base case. Applies to std::sort with dynamically load-balanced quicksort.
Referenced by __gnu_parallel::__qsb_local_sort_with_helping(), and set().

TLB_size

unsigned int __gnu_parallel::_Settings::TLB_size
size of the Translation Lookaside Buffer (underestimation).
Referenced by __gnu_parallel::__parallel_random_shuffle_drs(), __gnu_parallel::__sequential_random_shuffle(), and set().

transform_minimal_n

_SequenceIndex __gnu_parallel::_Settings::transform_minimal_n
Minimal input size for parallel std::transform.
Referenced by set().

unique_copy_minimal_n

_SequenceIndex __gnu_parallel::_Settings::unique_copy_minimal_n
Minimal input size for unique_copy.
Referenced by set().
The documentation for this struct was generated from the following file:

- settings.h

5.165 std::_Sp_ebo_helper< _Nm, _Tp, false > Struct Template Reference

#include <shared_ptr_base.h>

Public Member Functions

- **_Sp_ebo_helper** (_Tp &&__tp)
- **_Sp_ebo_helper** (const _Tp &__tp)

Static Public Member Functions

- static _Tp & **_S_get** (_Sp_ebo_helper &__eboh)

5.165.1 Detailed Description

template<int _Nm, typename _Tp>
struct std::_Sp_ebo_helper< _Nm, _Tp, false >

Specialization not using EBO.
The documentation for this struct was generated from the following file:

- shared_ptr_base.h

5.166 std::_Sp_ebo_helper< _Nm, _Tp, true > Struct Template Reference

`#include <shared_ptr_base.h>`

Public Member Functions

- **_Sp_ebo_helper** (_Tp &&__tp)
- **_Sp_ebo_helper** (const _Tp &__tp)

Static Public Member Functions

- static _Tp & **_S_get** (_Sp_ebo_helper &__eboh)

5.166.1 Detailed Description

template<int _Nm, typename _Tp>
struct std::_Sp_ebo_helper< _Nm, _Tp, true >

Specialization using EBO.
The documentation for this struct was generated from the following file:

- shared_ptr_base.h

5.167 __gnu_parallel::_SplitConsistently< __exact, _RAIter, _Compare, _SortingPlacesIterator > Struct Template Reference

`#include <multiway_mergesort.h>`

5.167.1 Detailed Description

template<bool __exact, typename _RAIter, typename _Compare, typename _SortingPlacesIterator>
struct __gnu_parallel::_SplitConsistently< __exact, _RAIter, _Compare, _SortingPlacesIterator >

Split consistently.
The documentation for this struct was generated from the following file:

- multiway_mergesort.h

5.168 __gnu_parallel::_SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIterator > Struct Template Reference

`#include <multiway_mergesort.h>`

Public Member Functions

- void **operator()** (const _ThreadIndex __iam, _PMWMSSortingData< _RAIter > *__sd, _Compare &__comp, const typename std::iterator_traits< _RAIter >::difference_type __num_samples) const

5.168.1 Detailed Description

template<typename _RAIter, typename _Compare, typename _SortingPlacesIterator>
struct __gnu_parallel::_SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIterator >

Split by sampling.
The documentation for this struct was generated from the following file:

- multiway_mergesort.h

5.169 __gnu_parallel::_SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator > Struct Template Reference

`#include <multiway_mergesort.h>`

Public Member Functions

- void **operator()** (const _ThreadIndex __iam, _PMWMSSortingData< _RAIter > *__sd, _Compare &__comp, const typename std::iterator_traits< _RAIter >::difference_type __num_samples) const

5.169.1 Detailed Description

template<typename _RAIter, typename _Compare, typename _SortingPlacesIterator>
struct __gnu_parallel::_SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator >

Split by exact splitting.
The documentation for this struct was generated from the following file:

- multiway_mergesort.h

5.170 std::__detail::_StateSeq< _TraitsT > Class Template Reference

`#include <regex_automaton.h>`

Public Types

- typedef _NFA< _TraitsT > **_RegexT**

Public Member Functions

- **_StateSeq** (_RegexT &__nfa, _StateIdT __s)
- **_StateSeq** (_RegexT &__nfa, _StateIdT __s, _StateIdT __end)
- void **_M_append** (_StateIdT __id)
- void **_M_append** (const _StateSeq &__s)
- _StateSeq **_M_clone** ()

Public Attributes

- _StateIdT **_M_end**
- _RegexT & **_M_nfa**
- _StateIdT **_M_start**

5.170.1 Detailed Description

template<typename _TraitsT>
class std::__detail::_StateSeq< _TraitsT >

Describes a sequence of one or more _State, its current start and end(s). This structure contains fragments of an NFA during construction.
The documentation for this class was generated from the following files:

- regex_automaton.h
- regex_automaton.tcc

5.171 __gnu_cxx::_Std_pointer_impl< _Tp > Class Template Reference

`#include <pointer.h>`

Public Types

- typedef _Tp **element_type**

Public Member Functions

- _Tp * **get** () const
- bool **operator<** (const _Std_pointer_impl &__rarg) const
- bool **operator==** (const _Std_pointer_impl &__rarg) const
- void **set** (element_type *__arg)

5.171.1 Detailed Description

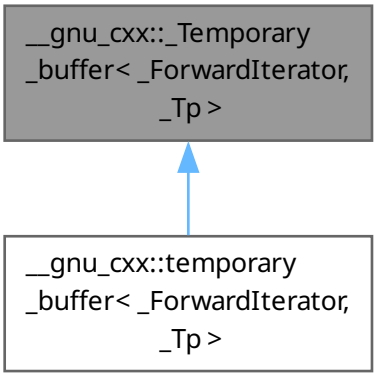
template<typename _Tp>
class __gnu_cxx::_Std_pointer_impl< _Tp >

A storage policy for use with _Pointer_adapter<> which yields a standard pointer.
A _Storage_policy is required to provide 4 things: 1) A get() API for returning the stored pointer value. 2) An set() API for storing a pointer value. 3) An element_type typedef to define the type this points to. 4) An operator<() to support pointer comparison. 5) An operator==() to support pointer comparison.
The documentation for this class was generated from the following file:

- pointer.h

5.172 __gnu_cxx::_Temporary_buffer< _ForwardIterator, _Tp > Class Template Reference

`#include <stl_tempbuf.h>`
Inheritance diagram for __gnu_cxx::_Temporary_buffer< _ForwardIterator, _Tp >:



Public Types

- typedef pointer **iterator**
- typedef value_type * **pointer**
- typedef ptrdiff_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- _Temporary_buffer (_ForwardIterator __seed, size_type __original_len)
- size_type _M_requested_size () const
- iterator begin ()
- iterator end ()
- size_type size () const

Protected Attributes

- struct std::_Temporary_buffer::_Impl **_M_impl**
- size_type **_M_original_len**

5.172.1 Detailed Description

template<typename _ForwardIterator, typename _Tp>
class __gnu_cxx::_Temporary_buffer< _ForwardIterator, _Tp >

This class is used in two places: stl_algo.h and ext/memory, where it is wrapped as the temporary_buffer class. See temporary_buffer docs for more notes.

5.172.2 Constructor & Destructor Documentation

_Temporary_buffer()

```
template<typename _ForwardIterator, typename _Tp>
std::_Temporary_buffer< _ForwardIterator, _Tp >::_Temporary_buffer (
            _ForwardIterator __seed,
            size_type __original_len)
```

Constructs a temporary buffer of a size somewhere between zero and the given length.

5.172.3 Member Function Documentation

_M_requested_size()

```
template<typename _ForwardIterator, typename _Tp>
size_type std::_Temporary_buffer< _ForwardIterator, _Tp >::_M_requested_size () const  [inline]
```

Returns the size requested by the constructor; may be >size().

begin()

```
template<typename _ForwardIterator, typename _Tp>
iterator std::_Temporary_buffer< _ForwardIterator, _Tp >::begin ()  [inline]
```

As per Table mumble.

end()

```
template<typename _ForwardIterator, typename _Tp>
iterator std::_Temporary_buffer< _ForwardIterator, _Tp >::end ()  [inline]
```

As per Table mumble.

size()

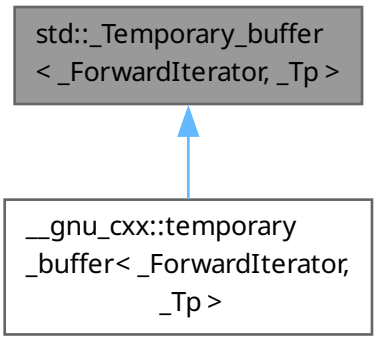
```
template<typename _ForwardIterator, typename _Tp>
size_type std::_Temporary_buffer< _ForwardIterator, _Tp >::size () const  [inline]
```

As per Table mumble.
The documentation for this class was generated from the following file:

- stl_tempbuf.h

5.173 std::_Temporary_buffer< _ForwardIterator, _Tp > Class Template Reference

`#include <stl_tempbuf.h>`
Inheritance diagram for std::_Temporary_buffer< _ForwardIterator, _Tp >:



Public Types

- typedef pointer **iterator**
- typedef value_type * **pointer**
- typedef ptrdiff_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- _Temporary_buffer (_ForwardIterator __seed, size_type __original_len)
- size_type _M_requested_size () const
- iterator begin ()
- iterator end ()
- size_type size () const

Protected Attributes

- struct std::_Temporary_buffer::_Impl **_M_impl**
- size_type **_M_original_len**

5.173.1 Detailed Description

template<typename _ForwardIterator, typename _Tp>
class std::_Temporary_buffer< _ForwardIterator, _Tp >

This class is used in two places: stl_algo.h and ext/memory, where it is wrapped as the temporary_buffer class. See temporary_buffer docs for more notes.

5.173.2 Constructor & Destructor Documentation

_Temporary_buffer()

```
template<typename _ForwardIterator, typename _Tp>
std::_Temporary_buffer< _ForwardIterator, _Tp >::_Temporary_buffer (
            _ForwardIterator __seed,
            size_type __original_len)
```

Constructs a temporary buffer of a size somewhere between zero and the given length.
References begin(), and end().

5.173.3 Member Function Documentation

_M_requested_size()

```
template<typename _ForwardIterator, typename _Tp>
size_type std::_Temporary_buffer< _ForwardIterator, _Tp >::_M_requested_size () const  [inline]
```

Returns the size requested by the constructor; may be >size().

begin()

```
template<typename _ForwardIterator, typename _Tp>
iterator std::_Temporary_buffer< _ForwardIterator, _Tp >::begin ()  [inline]
```

As per Table mumble.
Referenced by _Temporary_buffer().

end()

```
template<typename _ForwardIterator, typename _Tp>
iterator std::_Temporary_buffer< _ForwardIterator, _Tp >::end ()  [inline]
```

As per Table mumble.
Referenced by _Temporary_buffer().

size()

```
template<typename _ForwardIterator, typename _Tp>
size_type std::_Temporary_buffer< _ForwardIterator, _Tp >::size () const  [inline]
```

As per Table mumble.
The documentation for this class was generated from the following file:

- stl_tempbuf.h

5.174 __gnu_cxx::_Unqualified_type< _Tp > Struct Template Reference

`#include <pointer.h>`

Public Types

- typedef _Tp **type**

5.174.1 Detailed Description

template<typename _Tp>
struct __gnu_cxx::_Unqualified_type< _Tp >

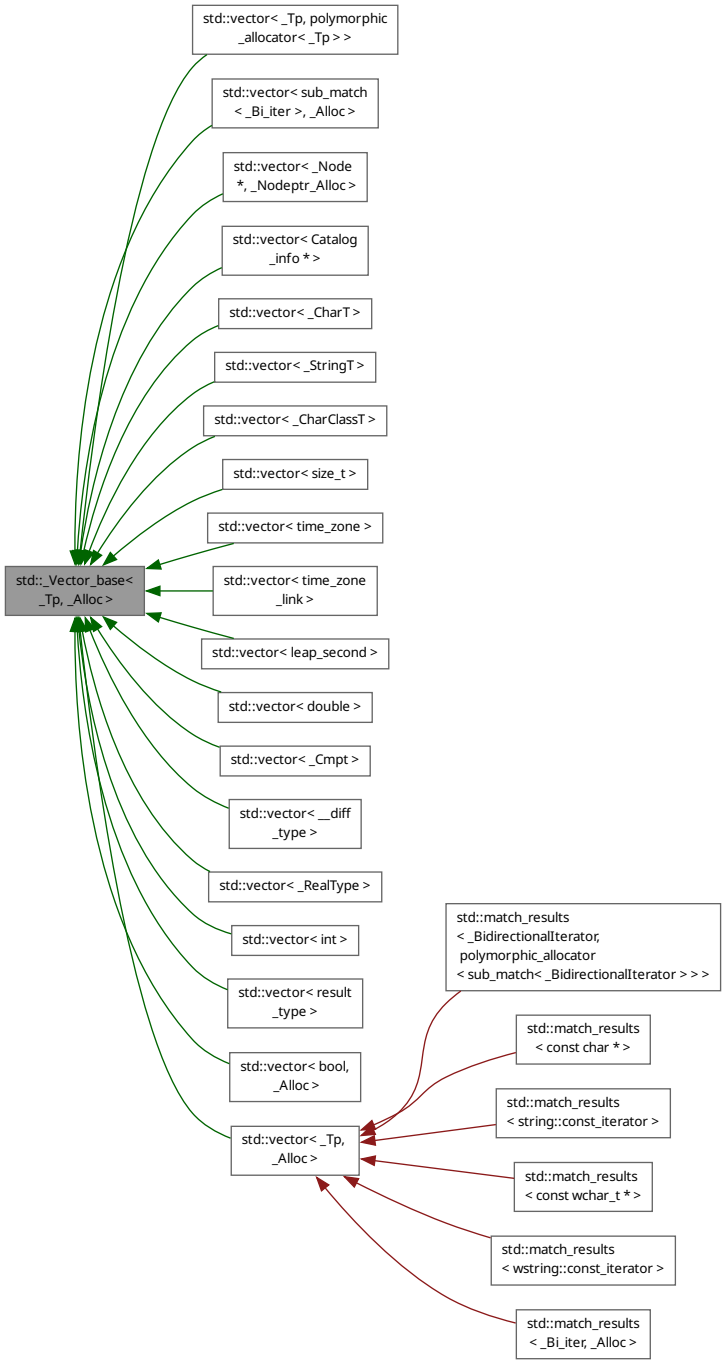
This structure accommodates the way in which std::iterator_traits<> is normally specialized for const T*, so that value_type is still T.
The documentation for this struct was generated from the following file:

- pointer.h

5.175 std::_Vector_base< _Tp, _Alloc > Struct Template Reference

`#include <stl_vector.h>`

Inheritance diagram for std::_Vector_base< _Tp, _Alloc >:



Public Types

- typedef __gnu_cxx::__alloc_traits< _Alloc >::template rebind< _Tp >::other **_Tp_alloc_type**
- typedef _Alloc **allocator_type**
- typedef __gnu_cxx::__alloc_traits< _Tp_alloc_type >::pointer **pointer**

Public Member Functions

- constexpr **_Vector_base** (_Tp_alloc_type &&__a) noexcept
- **_Vector_base** (_Vector_base &&)=default
- constexpr **_Vector_base** (_Vector_base &&__x, const allocator_type &__a)
- constexpr **_Vector_base** (const allocator_type &__a) noexcept
- constexpr **_Vector_base** (const allocator_type &__a, _Vector_base &&__x)
- constexpr **_Vector_base** (size_t __n)
- constexpr **_Vector_base** (size_t __n, const allocator_type &__a)
- constexpr pointer **_M_allocate** (size_t __n)
- constexpr void **_M_deallocate** (pointer __p, size_t __n)
- constexpr const _Tp_alloc_type & **_M_get_Tp_allocator** () const noexcept
- constexpr _Tp_alloc_type & **_M_get_Tp_allocator** () noexcept
- constexpr allocator_type **get_allocator** () const noexcept

Public Attributes

- _Vector_impl **_M_impl**

Protected Member Functions

- constexpr void **_M_create_storage** (size_t __n)

5.175.1 Detailed Description

template<**typename _Tp, typename _Alloc**>
struct std::_Vector_base< **_Tp, _Alloc** >

See bits/stl_deque.h's _Deque_base for an explanation.
The documentation for this struct was generated from the following file:

- stl_vector.h

5.176 std::add_const< _Tp > Struct Template Reference

`#include <type_traits>`

Public Types

- using **type**

5.176.1 Detailed Description

template<**typename _Tp**>
struct std::add_const< **_Tp** >

add_const
The documentation for this struct was generated from the following file:

- type_traits

5.177 std::add_cv< _Tp > Struct Template Reference

`#include <type_traits>`

Public Types

- using **type**

5.177.1 Detailed Description

template<typename _Tp>
struct std::add_cv< _Tp >

add_cv
The documentation for this struct was generated from the following file:

- type_traits

5.178 std::add_lvalue_reference< _Tp > Struct Template Reference

`#include <type_traits>`

Public Types

- using **type**

5.178.1 Detailed Description

template<typename _Tp>
struct std::add_lvalue_reference< _Tp >

add_lvalue_reference
The documentation for this struct was generated from the following file:

- type_traits

5.179 std::add_pointer< _Tp > Struct Template Reference

`#include <type_traits>`

Public Types

- using **type**

5.179.1 Detailed Description

template<typename _Tp>
struct std::add_pointer< _Tp >

add_pointer
The documentation for this struct was generated from the following file:

- type_traits

5.180 std::add_rvalue_reference< _Tp > Struct Template Reference

`#include <type_traits>`

Public Types

- using **type**

5.180.1 Detailed Description

template<typename _Tp>
struct std::add_rvalue_reference< _Tp >

add_rvalue_reference
The documentation for this struct was generated from the following file:

- type_traits

5.181 std::add_volatile< _Tp > Struct Template Reference

`#include <type_traits>`

Public Types

- using **type**

5.181.1 Detailed Description

template<typename _Tp>
struct std::add_volatile< _Tp >

add_volatile
The documentation for this struct was generated from the following file:

- type_traits

5.182 std::adopt_lock_t Struct Reference

`#include <std_mutex.h>`

5.182.1 Detailed Description

Assume the calling thread has already obtained mutex ownership and manage it.
The documentation for this struct was generated from the following file:

- std_mutex.h

5.183 std::aligned_storage< _Len, _Align > Struct Template Reference

`#include <type_traits>`

5.183.1 Detailed Description

template<size_t _Len, size_t _Align = __aligned_storage_default_alignment(_Len)>
struct std::aligned_storage< _Len, _Align >

Aligned storage.
The member typedef `type` is be a POD type suitable for use as uninitialized storage for any object whose size is at most `_Len` and whose alignment is a divisor of `_Align`.
It is important to use the nested `type` as uninitialized storage, not the `std::aligned_storage` type itself which is an empty class with 1-byte alignment. So this is correct:
`typename std::aligned_storage<sizeof(X), alignof(X)>::type m_xobj;`

This is wrong:
`std::aligned_storage<sizeof(X), alignof(X)> m_xobj;`
In C++14 and later `std::aligned_storage_t<sizeof(X), alignof(X)>` can be used to refer to the `type` member typedef.
The default value of _Align is supposed to be the most stringent fundamental alignment requirement for any C++ object type whose size is no greater than `_Len` (see [basic.align] in the C++ standard).

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

The documentation for this struct was generated from the following file:

- type_traits

5.184 std::aligned_union< _Len, _Types > Struct Template Reference

`#include <type_traits>`

Public Types

- using type

Static Public Attributes

- static const size_t alignment_value

5.184.1 Detailed Description

template<size_t _Len, typename... _Types>
struct std::aligned_union< _Len, _Types >

Provide aligned storage for types.
[meta.trans.other]
Provides aligned storage for any of the provided types of at least size _Len.

See also

aligned_storage

Deprecated Deprecated in C++23.

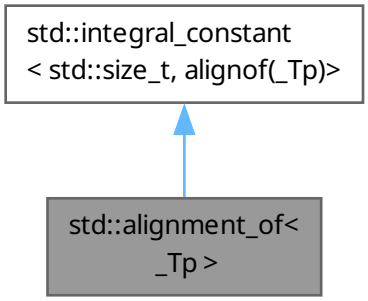
The documentation for this struct was generated from the following file:

- type_traits

5.185 std::alignment_of< _Tp > Struct Template Reference

`#include <type_traits>`

Inheritance diagram for std::alignment_of< _Tp >:



Public Types

- using **type**
- using **value_type**

Public Member Functions

- constexpr **operator value_type** () const noexcept

Static Public Attributes

- static constexpr std::size_t **value**

5.185.1 Detailed Description

template<**typename _Tp**>
struct std::alignment_of< **_Tp** >

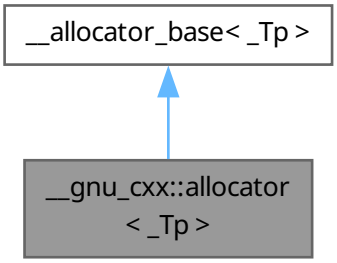
alignment_of
The documentation for this struct was generated from the following file:

- type_traits

5.186 __gnu_cxx::allocator< _Tp > Class Template Reference

```
#include <memory>
```

Inheritance diagram for __gnu_cxx::allocator< _Tp >:



Public Types

- typedef ptrdiff_t **difference_type**
- using **is_always_equal**
- using **propagate_on_container_move_assignment**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- constexpr **allocator** (const allocator &__a) noexcept
- template<typename _Tp1>
  constexpr **allocator** (const allocator< _Tp1 > &) noexcept
- constexpr _Tp * **allocate** (size_t __n)
- _Tp * **allocate** (size_type __n, const void *=static_cast< const void * >(0))
- constexpr void **deallocate** (_Tp *__p, size_t __n)
- allocator & **operator=** (const allocator &)=default

Friends

- constexpr bool **operator==** (const allocator &, const allocator &) noexcept

Related Symbols

(Note that these are not member symbols.)

- template<typename _T1, typename _T2>
  constexpr bool operator== (const allocator< _T1 > &, const allocator< _T2 > &) noexcept

5.186.1 Detailed Description

template<typename _Tp>
class __gnu_cxx::allocator< _Tp >

The *standard* allocator, as per C++03 [20.4.1].
See https://gcc.gnu.org/onlinedocs/libstdc++/manual/memory.html#std.util.memory.allocator for further details.

Template Parameters

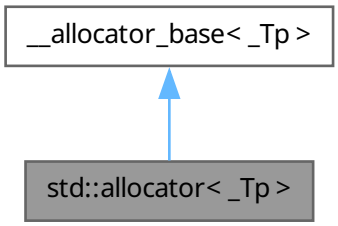
_Tp	Type of allocated object.

The documentation for this class was generated from the following file:

- allocator.h

5.187 std::allocator< _Tp > Class Template Reference

`#include <memory>`
Inheritance diagram for std::allocator< _Tp >:



Public Types

- typedef ptrdiff_t **difference_type**
- using **is_always_equal**
- using **propagate_on_container_move_assignment**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- constexpr **allocator** (const allocator &__a) noexcept
- template<typename _Tp1>
  constexpr **allocator** (const allocator< _Tp1 > &) noexcept
- _Tp * **allocate** (size_type __n, const void *=static_cast< const void * >(0))
- constexpr _Tp * **allocate** (size_t __n)
- constexpr void **deallocate** (_Tp *__p, size_t __n)
- allocator & **operator=** (const allocator &)=default

Friends

- constexpr bool **operator==** (const allocator &, const allocator &) noexcept

Related Symbols

(Note that these are not member symbols.)

- template<typename _T1, typename _T2>
  constexpr bool operator== (const allocator< _T1 > &, const allocator< _T2 > &) noexcept

5.187.1 Detailed Description

template<typename _Tp>
class std::allocator< _Tp >

The *standard* allocator, as per C++03 [20.4.1].
See https://gcc.gnu.org/onlinedocs/libstdc++/manual/memory.html#std.util.memory.allocator for further details.

Template Parameters

_Tp	Type of allocated object.

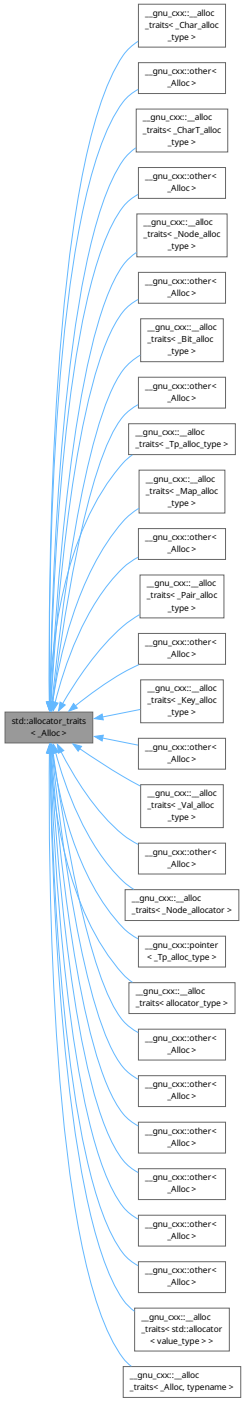
The documentation for this class was generated from the following file:

- allocator.h

5.188 std::allocator_traits< _Alloc > Struct Template Reference

`#include <memory>`

Inheritance diagram for std::allocator_traits< _Alloc >:



Public Types

- typedef _Alloc allocator_type
- using const_pointer

- using const_void_pointer
- using difference_type
- using is_always_equal
- using pointer
- using propagate_on_container_copy_assignment
- using propagate_on_container_move_assignment
- using propagate_on_container_swap
- template<typename _Tp>
  using **rebind_alloc**
- template<typename _Tp>
  using **rebind_traits**
- using size_type
- typedef _Alloc::value_type value_type
- using void_pointer

Static Public Member Functions

- static constexpr pointer allocate (_Alloc &__a, size_type __n)
- static constexpr pointer allocate (_Alloc &__a, size_type __n, const_void_pointer __hint)
- template<typename _Tp, typename... _Args>
  requires __can_construct<_Alloc, _Tp, _Args...>
  static constexpr void construct (_Alloc &__a, _Tp *__p, _Args &&... __args) noexcept(_S_nothrow_construct< _Tp, _Args... >())
- static constexpr void deallocate (_Alloc &__a, pointer __p, size_type __n)
- template<typename _Tp>
  static constexpr void destroy (_Alloc &__a, _Tp *__p) noexcept(_S_nothrow_destroy< _Tp >())
- static constexpr size_type max_size (const _Alloc &__a) noexcept
- static constexpr _Alloc select_on_container_copy_construction (const _Alloc &__rhs)

5.188.1 Detailed Description

template<typename _Alloc>
struct std::allocator_traits< _Alloc >

Uniform interface to all allocator types.

Since

C++11

5.188.2 Member Typedef Documentation

allocator_type

```
template<typename _Alloc>
typedef _Alloc std::allocator_traits< _Alloc >::allocator_type
```

The allocator type.

const_pointer

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::const_pointer
```

The allocator's const pointer type.
`Alloc::const_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<const value_type>`

const_void_pointer

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::const_void_pointer
```

The allocator's const void pointer type.
`Alloc::const_void_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<const void>`

difference_type

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::difference_type
```

The allocator's difference type.
`Alloc::difference_type` if that type exists, otherwise `pointer_traits<pointer>::difference_type`

is_always_equal

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::is_always_equal
```

Whether all instances of the allocator type compare equal.
`Alloc::is_always_equal` if that type exists, otherwise `is_empty<Alloc>::type`

pointer

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::pointer
```

The allocator's pointer type.
`Alloc::pointer` if that type exists, otherwise `value_type*`

propagate_on_container_copy_assignment

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::propagate_on_container_copy_assignment
```

How the allocator is propagated on copy assignment.
`Alloc::propagate_on_container_copy_assignment` if that type exists, otherwise `false_type`

propagate_on_container_move_assignment

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::propagate_on_container_move_assignment
```

How the allocator is propagated on move assignment.
`Alloc::propagate_on_container_move_assignment` if that type exists, otherwise `false_type`

propagate_on_container_swap

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::propagate_on_container_swap
```

How the allocator is propagated on swap.
`Alloc::propagate_on_container_swap` if that type exists, otherwise `false_type`

size_type

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::size_type
```

The allocator's size type.

`Alloc::size_type` if that type exists, otherwise `make_unsigned<difference_type>::type`

value_type

```
template<typename _Alloc>
typedef _Alloc::value_type std::allocator_traits< _Alloc >::value_type
```

The allocated type.

void_pointer

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::void_pointer
```

The allocator's void pointer type.

`Alloc::void_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<void>`

5.188.3 Member Function Documentation

allocate() [1/2]

```
template<typename _Alloc>
static constexpr pointer std::allocator_traits< _Alloc >::allocate (
            _Alloc & __a,
            size_type __n)  [inline], [static], [nodiscard], [constexpr]
```

Allocate memory.

Parameters

_↩ _a	An allocator.
_↩ _n	The number of objects to allocate space for.

Calls `a.allocate(n)`

allocate() [2/2]

```
template<typename _Alloc>
static constexpr pointer std::allocator_traits< _Alloc >::allocate (
            _Alloc & __a,
            size_type __n,
            const_void_pointer __hint)  [inline], [static], [nodiscard], [constexpr]
```

Allocate memory.

Parameters

__a	An allocator.
__n	The number of objects to allocate space for.
__hint	Aid to locality.

Returns

Memory of suitable size and alignment for *n* objects of type `value_type`

Returns `a.allocate(n, hint)` if that expression is well-formed, otherwise returns `a.allocate(n)`

construct()

```
template<typename _Alloc>
template<typename _Tp, typename... _Args>
requires __can_construct<_Alloc, _Tp, _Args...>
static constexpr void std::allocator_traits< _Alloc >::construct (
            _Alloc & __a,
            _Tp * __p,
            _Args &&... __args) [inline], [static], [constexpr], [noexcept]
```

Construct an object of type `_Tp`

Parameters

__a	An allocator.
__p	Pointer to memory of suitable size and alignment for Tp
__args	Constructor arguments.

Calls `__a.construct(__p, std::forward<Args>(__args)...)` if that expression is well-formed, otherwise uses placement-new to construct an object of type *_Tp* at location *__p* from the arguments *__args*...

deallocate()

```
template<typename _Alloc>
static constexpr void std::allocator_traits< _Alloc >::deallocate (
            _Alloc & __a,
            pointer __p,
            size_type __n) [inline], [static], [constexpr]
```

Deallocate memory.

Parameters

_↩ _a	An allocator.
_↩ _p	Pointer to the memory to deallocate.
_↩ _n	The number of objects space was allocated for.

Calls `a.deallocate(p, n)`

destroy()

```
template<typename _Alloc>
template<typename _Tp>
static constexpr void std::allocator_traits< _Alloc >::destroy (
            _Alloc & __a,
            _Tp * __p) [inline], [static], [constexpr], [noexcept]
```

Destroy an object of type *_Tp*.

Parameters

_↩ _a	An allocator.

_↩ _p	Pointer to the object to destroy

Calls __a.destroy(__p) if that expression is well-formed, otherwise calls __p->~_Tp()

max_size()

```
template<typename _Alloc>
static constexpr size_type std::allocator_traits< _Alloc >::max_size (
            const _Alloc & __a)  [inline], [static], [constexpr], [noexcept]
```

The maximum supported allocation size.

Parameters

_↩ _a	An allocator.

Returns

__a.max_size() or numeric_limits<size_type>::max()

Returns __a.max_size() if that expression is well-formed, otherwise returns numeric_limits<size_↩ type>::max()

select_on_container_copy_construction()

```
template<typename _Alloc>
static constexpr _Alloc std::allocator_traits< _Alloc >::select_on_container_copy_construction (
            const _Alloc & __rhs)  [inline], [static], [constexpr]
```

Obtain an allocator to use when copying a container.

Parameters

__rhs	An allocator.

Returns

__rhs.select_on_container_copy_construction() or *__rhs*

Returns __rhs.select_on_container_copy_construction() if that expression is well-formed, otherwise returns *__rhs*

The documentation for this struct was generated from the following file:

- bits/alloc_traits.h

5.189 std::allocator_traits< allocator< _Tp > > Struct Template Reference

#include <memory>

Public Types

- typedef allocator< _Tp > allocator_type
- using allocator_type
- using const_pointer
- using const_pointer
- using const_void_pointer
- using const_void_pointer
- using difference_type
- using difference_type
- using is_always_equal
- using is_always_equal
- using pointer
- using pointer
- using propagate_on_container_copy_assignment
- using propagate_on_container_copy_assignment
- using propagate_on_container_move_assignment
- using propagate_on_container_move_assignment
- using propagate_on_container_swap
- using propagate_on_container_swap
- using **rebind_alloc**
- template<typename _Up>
  using **rebind_alloc**
- using **rebind_traits**
- template<typename _Up>
  using **rebind_traits**
- using size_type
- using size_type
- typedef allocator< _Tp >::value_type value_type
- using value_type
- using void_pointer
- using void_pointer

Static Public Member Functions

- static constexpr pointer allocate (allocator< _Tp > &__a, size_type __n)
- static constexpr pointer allocate (allocator< _Tp > &__a, size_type __n, const_void_pointer __hint)
- static constexpr pointer allocate (allocator_type &__a, size_type __n)
- static constexpr pointer allocate (allocator_type &__a, size_type __n, const_void_pointer __hint)
- static constexpr void construct (allocator< _Tp > &__a, _Tp *__p, _Args &&... __args) noexcept(_S_nothrow↩ _construct< _Tp, _Args... >())
- template<typename _Up, typename... _Args>
  static constexpr void construct (allocator_type &__a, _Up *__p, _Args &&... __args) noexcept(__is_nothrow_↩ new_constructible< _Up, _Args... >)
- static constexpr void deallocate (allocator< _Tp > &__a, pointer __p, size_type __n)
- static constexpr void deallocate (allocator_type &__a, pointer __p, size_type __n)
- static constexpr void destroy (allocator< _Tp > &__a, _Tp *__p) noexcept(_S_nothrow_destroy< _Tp >())
- template<typename _Up>
  static constexpr void destroy (allocator_type &__a, _Up *__p) noexcept(is_nothrow_destructible< _Up >::value)
- static constexpr size_type max_size (const allocator< _Tp > &__a) noexcept
- static constexpr size_type max_size (const allocator_type &__a) noexcept
- static constexpr allocator< _Tp > select_on_container_copy_construction (const allocator< _Tp > &__rhs)
- static constexpr allocator_type select_on_container_copy_construction (const allocator_type &__rhs)

5.189.1 Detailed Description

template<typename _Tp>
struct std::allocator_traits< allocator< _Tp > >

Partial specialization for `std::allocator`

Since

C++11

See also

std::allocator_traits

5.189.2 Member Typedef Documentation

allocator_type [1/2]

```
typedef allocator< _Tp > std::allocator_traits< allocator< _Tp > >::allocator_type
```

The allocator type.

allocator_type [2/2]

```
template<typename _Tp>
using std::allocator_traits< allocator< _Tp > >::allocator_type
```

The allocator type.

const_pointer [1/2]

```
using std::allocator_traits< allocator< _Tp > >::const_pointer
```

The allocator's const pointer type.
`Alloc::const_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<const value_type>`

const_pointer [2/2]

```
template<typename _Tp>
using std::allocator_traits< allocator< _Tp > >::const_pointer
```

The allocator's const pointer type.

const_void_pointer [1/2]

```
using std::allocator_traits< allocator< _Tp > >::const_void_pointer
```

The allocator's const void pointer type.
`Alloc::const_void_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<const void>`

const_void_pointer [2/2]

```
template<typename _Tp>
using std::allocator_traits< allocator< _Tp > >::const_void_pointer
```

The allocator's const void pointer type.

difference_type [1/2]

```
using std::allocator_traits< allocator< _Tp > >::difference_type
```

The allocator's difference type.
`Alloc::difference_type` if that type exists, otherwise `pointer_traits<pointer>::difference_type`

difference_type [2/2]

```
template<typename _Tp>
using std::allocator_traits< allocator< _Tp > >::difference_type
```

The allocator's difference type.

is_always_equal [1/2]

```
using std::allocator_traits< allocator< _Tp > >::is_always_equal
```

Whether all instances of the allocator type compare equal.
`Alloc::is_always_equal` if that type exists, otherwise `is_empty<Alloc>::type`

is_always_equal [2/2]

```
template<typename _Tp>
using std::allocator_traits< allocator< _Tp > >::is_always_equal
```

Whether all instances of the allocator type compare equal.

pointer [1/2]

```
using std::allocator_traits< allocator< _Tp > >::pointer
```

The allocator's pointer type.
`Alloc::pointer` if that type exists, otherwise `value_type*`

pointer [2/2]

```
template<typename _Tp>
using std::allocator_traits< allocator< _Tp > >::pointer
```

The allocator's pointer type.

propagate_on_container_copy_assignment [1/2]

```
using std::allocator_traits< allocator< _Tp > >::propagate_on_container_copy_assignment
```

How the allocator is propagated on copy assignment.
`Alloc::propagate_on_container_copy_assignment` if that type exists, otherwise `false_type`

propagate_on_container_copy_assignment [2/2]

```
template<typename _Tp>
using std::allocator_traits< allocator< _Tp > >::propagate_on_container_copy_assignment
```

How the allocator is propagated on copy assignment.

propagate_on_container_move_assignment [1/2]

```
using std::allocator_traits< allocator< _Tp > >::propagate_on_container_move_assignment
```

How the allocator is propagated on move assignment.
`Alloc::propagate_on_container_move_assignment` if that type exists, otherwise `false_type`

propagate_on_container_move_assignment [2/2]

```
template<typename _Tp>
using std::allocator_traits< allocator< _Tp > >::propagate_on_container_move_assignment
```

How the allocator is propagated on move assignment.

propagate_on_container_swap [1/2]

```
using std::allocator_traits< allocator< _Tp > >::propagate_on_container_swap
```

How the allocator is propagated on swap.

`Alloc::propagate_on_container_swap` if that type exists, otherwise `false_type`

propagate_on_container_swap [2/2]

```
template<typename _Tp>
using std::allocator_traits< allocator< _Tp > >::propagate_on_container_swap
```

How the allocator is propagated on swap.

size_type [1/2]

```
using std::allocator_traits< allocator< _Tp > >::size_type
```

The allocator's size type.

`Alloc::size_type` if that type exists, otherwise `make_unsigned<difference_type>::type`

size_type [2/2]

```
template<typename _Tp>
using std::allocator_traits< allocator< _Tp > >::size_type
```

The allocator's size type.

value_type [1/2]

```
typedef allocator< _Tp >::value_type std::allocator_traits< allocator< _Tp > >::value_type
```

The allocated type.

value_type [2/2]

```
template<typename _Tp>
using std::allocator_traits< allocator< _Tp > >::value_type
```

The allocated type.

void_pointer [1/2]

```
using std::allocator_traits< allocator< _Tp > >::void_pointer
```

The allocator's void pointer type.

`Alloc::void_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<void>`

void_pointer [2/2]

```
template<typename _Tp>
using std::allocator_traits< allocator< _Tp > >::void_pointer
```

The allocator's void pointer type.

5.189.3 Member Function Documentation

allocate() [1/4]

```
static constexpr pointer std::allocator_traits< allocator< _Tp > >::allocate (
            allocator< _Tp > & __a,
            size_type __n)  [inline], [static], [nodiscard], [constexpr]
```

Allocate memory.

Parameters

_↩ _a	An allocator.
_↩ _n	The number of objects to allocate space for.

Calls `a.allocate(n)`

allocate() [2/4]

```
static constexpr pointer std::allocator_traits< allocator< _Tp > >::allocate (
            allocator< _Tp > & __a,
            size_type __n,
            const_void_pointer __hint)  [inline], [static], [nodiscard], [constexpr]
```

Allocate memory.

Parameters

__a	An allocator.
__n	The number of objects to allocate space for.
__hint	Aid to locality.

Returns

Memory of suitable size and alignment for *n* objects of type `value_type`

Returns `a.allocate(n, hint)` if that expression is well-formed, otherwise returns `a.allocate(n)`

allocate() [3/4]

```
template<typename _Tp>
static constexpr pointer std::allocator_traits< allocator< _Tp > >::allocate (
            allocator_type & __a,
            size_type __n)  [inline], [static], [constexpr]
```

Allocate memory.

Parameters

_↩ _a	An allocator.
_↩ _n	The number of objects to allocate space for.

Calls `a.allocate(n)`

allocate() [4/4]

```
template<typename _Tp>
static constexpr pointer std::allocator_traits< allocator< _Tp > >::allocate (
            allocator_type & __a,
            size_type __n,
            const_void_pointer __hint)  [inline], [static], [constexpr]
```

Allocate memory.

Parameters

__a	An allocator.
__n	The number of objects to allocate space for.
__hint	Aid to locality.

Returns

Memory of suitable size and alignment for *n* objects of type `value_type`

Returns `a.allocate(n, hint)`

construct() [1/2]

```
static constexpr void std::allocator_traits< allocator< _Tp > >::construct (
            allocator< _Tp > & __a,
            _Tp * __p,
            _Args &&... __args)  [inline], [static], [constexpr], [noexcept]
```

Construct an object of type `_Tp`

Parameters

__a	An allocator.
__p	Pointer to memory of suitable size and alignment for Tp
__args	Constructor arguments.

Calls `__a.construct(__p, std::forward<Args>(__args)...)` if that expression is well-formed, otherwise uses placement-new to construct an object of type *_Tp* at location *__p* from the arguments *__args*...

construct() [2/2]

```
template<typename _Tp>
template<typename _Up, typename... _Args>
static constexpr void std::allocator_traits< allocator< _Tp > >::construct (
            allocator_type & __a,
            _Up * __p,
            _Args &&... __args)  [inline], [static], [constexpr], [noexcept]
```

Construct an object of type `_Up`

Parameters

__a	An allocator.
__p	Pointer to memory of suitable size and alignment for an object of type `_Up`.
__args	Constructor arguments.

Calls `__a.construct(__p, std::forward<_Args>(__args)...)` in C++11, C++14 and C++17. Changed in C++20 to call `std::construct_at(__p, std::forward<_Args>(__args)...)` instead. References std::_Construct(), and std::forward().

deallocate() [1/2]

```
static constexpr void std::allocator_traits< allocator< _Tp > >::deallocate (
            allocator< _Tp > & __a,
            pointer __p,
            size_type __n)  [inline], [static], [constexpr]
```

Deallocate memory.

Parameters

_↩ _a	An allocator.
_↩ _p	Pointer to the memory to deallocate.
_↩ _n	The number of objects space was allocated for.

Calls `a.deallocate(p, n)`

deallocate() [2/2]

```
template<typename _Tp>
static constexpr void std::allocator_traits< allocator< _Tp > >::deallocate (
            allocator_type & __a,
            pointer __p,
            size_type __n)  [inline], [static], [constexpr]
```

Deallocate memory.

Parameters

_↩ _a	An allocator.
_↩ _p	Pointer to the memory to deallocate.
_↩ _n	The number of objects space was allocated for.

Calls `a.deallocate(p, n)`

destroy() [1/2]

```
static constexpr void std::allocator_traits< allocator< _Tp > >::destroy (
            allocator< _Tp > & __a,
            _Tp * __p)  [inline], [static], [constexpr], [noexcept]
```

Destroy an object of type *_Tp*.

Parameters

_↩ _a	An allocator.
_↩ _p	Pointer to the object to destroy

Calls __a.destroy(__p) if that expression is well-formed, otherwise calls __p->~_Tp()

destroy() [2/2]

```
template<typename _Tp>
template<typename _Up>
static constexpr void std::allocator_traits< allocator< _Tp > >::destroy (
            allocator_type & __a,
            _Up * __p)  [inline], [static], [constexpr], [noexcept]
```

Destroy an object of type *_Up*.

Parameters

_↩ _a	An allocator.
_↩ _p	Pointer to the object to destroy

Calls __a.destroy(__p).

max_size() [1/2]

```
static constexpr size_type std::allocator_traits< allocator< _Tp > >::max_size (
            const allocator< _Tp > & __a)  [inline], [static], [constexpr], [noexcept]
```

The maximum supported allocation size.

Parameters

_↩ _a	An allocator.

Returns

__a.max_size() or numeric_limits<size_type>::max()

Returns __a.max_size() if that expression is well-formed, otherwise returns numeric_limits<size_↩ type>::max()

max_size() [2/2]

```
template<typename _Tp>
static constexpr size_type std::allocator_traits< allocator< _Tp > >::max_size (
            const allocator_type & __a)  [inline], [static], [constexpr], [noexcept]
```

The maximum supported allocation size.

Parameters

_↩ _a	An allocator.

Returns

`__a.max_size()`

select_on_container_copy_construction() [1/2]

```
static constexpr allocator< _Tp > std::allocator_traits< allocator< _Tp > >::select_on_container↩
_copy_construction (
            const allocator< _Tp > & __rhs)  [inline], [static], [constexpr]
```

Obtain an allocator to use when copying a container.

Parameters

__rhs	An allocator.

Returns

`__rhs.select_on_container_copy_construction()` or *__rhs*

Returns `__rhs.select_on_container_copy_construction()` if that expression is well-formed, otherwise returns *__rhs*

select_on_container_copy_construction() [2/2]

```
template<typename _Tp>
static constexpr allocator_type std::allocator_traits< allocator< _Tp > >::select_on_container↩
_copy_construction (
            const allocator_type & __rhs)  [inline], [static], [constexpr]
```

Obtain an allocator to use when copying a container.

Parameters

__rhs	An allocator.

Returns

`__rhs`

The documentation for this struct was generated from the following file:

- bits/alloc_traits.h

5.190 std::allocator_traits< allocator< void > > Struct Reference

`#include <memory>`

Public Types

- typedef allocator< void > allocator_type
- using allocator_type
- using const_pointer
- using const_pointer
- using const_void_pointer
- using const_void_pointer
- using difference_type
- using difference_type
- using is_always_equal
- using is_always_equal
- using pointer
- using pointer
- using propagate_on_container_copy_assignment
- using propagate_on_container_copy_assignment
- using propagate_on_container_move_assignment
- using propagate_on_container_move_assignment
- using propagate_on_container_swap
- using propagate_on_container_swap
- using **rebind_alloc**
- template<typename _Up>
  using **rebind_alloc**
- using **rebind_traits**
- template<typename _Up>
  using **rebind_traits**
- using size_type
- using size_type
- typedef allocator< void >::value_type value_type
- using value_type
- using void_pointer
- using void_pointer

Static Public Member Functions

- static constexpr pointer allocate (allocator< void > &__a, size_type __n)
- static constexpr pointer allocate (allocator< void > &__a, size_type __n, const_void_pointer __hint)
- static void * allocate (allocator_type &, size_type, const void *=nullptr)=delete
- static constexpr void construct (allocator< void > &__a, _Tp *__p, _Args &&... __args) noexcept(_S_nothrow↩ _construct< _Tp, _Args... >())
- template<typename _Up, typename... _Args>
  static constexpr void construct (allocator_type &, _Up *__p, _Args &&... __args) noexcept(__is_nothrow_new↩ _constructible< _Up, _Args... >)
- static constexpr void deallocate (allocator< void > &__a, pointer __p, size_type __n)
- static void deallocate (allocator_type &, void *, size_type)=delete
- static constexpr void destroy (allocator< void > &__a, _Tp *__p) noexcept(_S_nothrow_destroy< _Tp >())
- template<typename _Up>
  static constexpr void destroy (allocator_type &, _Up *__p) noexcept(is_nothrow_destructible< _Up >::value)
- static constexpr size_type max_size (const allocator< void > &__a) noexcept
- static size_type max_size (const allocator_type &)=delete
- static constexpr allocator< void > select_on_container_copy_construction (const allocator< void > &__rhs)
- static constexpr allocator_type select_on_container_copy_construction (const allocator_type &__rhs)

5.190.1 Detailed Description

Explicit specialization for `std::allocator<void>`

Since

C++11

See also

std::allocator_traits

5.190.2 Member Typedef Documentation

allocator_type [1/2]

```
typedef allocator< void > std::allocator_traits< allocator< void > >::allocator_type
```

The allocator type.

allocator_type [2/2]

```
using std::allocator_traits< allocator< void > >::allocator_type
```

The allocator type.

const_pointer [1/2]

```
using std::allocator_traits< allocator< void > >::const_pointer
```

The allocator's const pointer type.
`Alloc::const_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<const value_type>`

const_pointer [2/2]

```
using std::allocator_traits< allocator< void > >::const_pointer
```

The allocator's const pointer type.

const_void_pointer [1/2]

```
using std::allocator_traits< allocator< void > >::const_void_pointer
```

The allocator's const void pointer type.
`Alloc::const_void_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<const void>`

const_void_pointer [2/2]

```
using std::allocator_traits< allocator< void > >::const_void_pointer
```

The allocator's const void pointer type.

difference_type [1/2]

```
using std::allocator_traits< allocator< void > >::difference_type
```

The allocator's difference type.
`Alloc::difference_type` if that type exists, otherwise `pointer_traits<pointer>::difference_type`

difference_type [2/2]

```
using std::allocator_traits< allocator< void > >::difference_type
```

The allocator's difference type.

is_always_equal [1/2]

```
using std::allocator_traits< allocator< void > >::is_always_equal
```

Whether all instances of the allocator type compare equal.
`Alloc::is_always_equal` if that type exists, otherwise `is_empty<Alloc>::type`

is_always_equal [2/2]

```
using std::allocator_traits< allocator< void > >::is_always_equal
```

Whether all instances of the allocator type compare equal.

pointer [1/2]

```
using std::allocator_traits< allocator< void > >::pointer
```

The allocator's pointer type.
`Alloc::pointer` if that type exists, otherwise `value_type*`

pointer [2/2]

```
using std::allocator_traits< allocator< void > >::pointer
```

The allocator's pointer type.

propagate_on_container_copy_assignment [1/2]

```
using std::allocator_traits< allocator< void > >::propagate_on_container_copy_assignment
```

How the allocator is propagated on copy assignment.
`Alloc::propagate_on_container_copy_assignment` if that type exists, otherwise `false_type`

propagate_on_container_copy_assignment [2/2]

```
using std::allocator_traits< allocator< void > >::propagate_on_container_copy_assignment
```

How the allocator is propagated on copy assignment.

propagate_on_container_move_assignment [1/2]

```
using std::allocator_traits< allocator< void > >::propagate_on_container_move_assignment
```

How the allocator is propagated on move assignment.
`Alloc::propagate_on_container_move_assignment` if that type exists, otherwise `false_type`

propagate_on_container_move_assignment [2/2]

```
using std::allocator_traits< allocator< void > >::propagate_on_container_move_assignment
```

How the allocator is propagated on move assignment.

propagate_on_container_swap [1/2]

```
using std::allocator_traits< allocator< void > >::propagate_on_container_swap
```

How the allocator is propagated on swap.
`Alloc::propagate_on_container_swap` if that type exists, otherwise `false_type`

propagate_on_container_swap [2/2]

```
using std::allocator_traits< allocator< void > >::propagate_on_container_swap
```

How the allocator is propagated on swap.

size_type [1/2]

```
using std::allocator_traits< allocator< void > >::size_type
```

The allocator's size type.
`Alloc::size_type` if that type exists, otherwise `make_unsigned<difference_type>::type`

size_type [2/2]

```
using std::allocator_traits< allocator< void > >::size_type
```

The allocator's size type.

value_type [1/2]

```
typedef allocator< void >::value_type std::allocator_traits< allocator< void > >::value_type
```

The allocated type.

value_type [2/2]

```
using std::allocator_traits< allocator< void > >::value_type
```

The allocated type.

void_pointer [1/2]

```
using std::allocator_traits< allocator< void > >::void_pointer
```

The allocator's void pointer type.
`Alloc::void_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<void>`

void_pointer [2/2]

```
using std::allocator_traits< allocator< void > >::void_pointer
```

The allocator's void pointer type.

5.190.3 Member Function Documentation

allocate() [1/3]

```
static constexpr pointer std::allocator_traits< allocator< void > >::allocate (
            allocator< void > & __a,
            size_type __n)  [inline], [static], [nodiscard], [constexpr]
```

Allocate memory.

Parameters

__a	An allocator.
__n	The number of objects to allocate space for.

Calls `a.allocate(n)`

allocate() [2/3]

```
static constexpr pointer std::allocator_traits< allocator< void > >::allocate (
            allocator< void > & __a,
            size_type __n,
            const_void_pointer __hint)  [inline], [static], [nodiscard], [constexpr]
```

Allocate memory.

Parameters

__a	An allocator.
__n	The number of objects to allocate space for.
__hint	Aid to locality.

Returns

Memory of suitable size and alignment for *n* objects of type `value_type`

Returns `a.allocate(n, hint)` if that expression is well-formed, otherwise returns `a.allocate(n)`

allocate() [3/3]

```
static void * std::allocator_traits< allocator< void > >::allocate (
            allocator_type & ,
            size_type ,
            const void *  = nullptr)  [static], [delete]
```

allocate is ill-formed for allocator<void>

construct() [1/2]

```
static constexpr void std::allocator_traits< allocator< void > >::construct (
            allocator< void > & __a,
            _Tp * __p,
            _Args &&...  __args)  [inline], [static], [constexpr], [noexcept]
```

Construct an object of type `_Tp`

Parameters

__a	An allocator.
__p	Pointer to memory of suitable size and alignment for Tp
__args	Constructor arguments.

Calls `__a.construct(__p, std::forward<Args>(__args)...)` if that expression is well-formed, otherwise uses placement-new to construct an object of type *_Tp* at location *__p* from the arguments *__args*...

construct() [2/2]

```
template<typename _Up, typename...  _Args>
static constexpr void std::allocator_traits< allocator< void > >::construct (
            allocator_type & ,
            _Up * __p,
            _Args &&...  __args)  [inline], [static], [constexpr], [noexcept]
```

Construct an object of type `_Up`

Parameters

__a	An allocator.
__p	Pointer to memory of suitable size and alignment for an object of type `_Up`.
__args	Constructor arguments.

Calls `__a.construct(__p, std::forward<_Args>(__args)...)` in C++11, C++14 and C++17. Changed in C++20 to call `std::construct_at(__p, std::forward<_Args>(__args)...)` instead.
References std::_Construct(), and std::forward().

deallocate() [1/2]

```
static constexpr void std::allocator_traits< allocator< void > >::deallocate (
            allocator< void > & __a,
            pointer __p,
            size_type __n)  [inline], [static], [constexpr]
```

Deallocate memory.

Parameters

_↩ _a	An allocator.
_↩ _p	Pointer to the memory to deallocate.
_↩ _n	The number of objects space was allocated for.

Calls `a.deallocate(p, n)`

deallocate() [2/2]

```
static void std::allocator_traits< allocator< void > >::deallocate (
            allocator_type & ,
            void * ,
            size_type )  [static], [delete]
```

deallocate is ill-formed for allocator<void>

destroy() [1/2]

```
static constexpr void std::allocator_traits< allocator< void > >::destroy (
            allocator< void > & __a,
            _Tp * __p)  [inline], [static], [constexpr], [noexcept]
```

Destroy an object of type *_Tp*.

Parameters

_↩ _a	An allocator.
_↩ _p	Pointer to the object to destroy

Calls `__a.destroy(__p)` if that expression is well-formed, otherwise calls `__p->~_Tp()`

destroy() [2/2]

```
template<typename _Up>
static constexpr void std::allocator_traits< allocator< void > >::destroy (
            allocator_type & ,
            _Up * __p)  [inline], [static], [constexpr], [noexcept]
```

Destroy an object of type _Up

Parameters

_↩ _a	An allocator.

Parameters

_↩ _p	Pointer to the object to destroy

Invokes the destructor for *__p.
References std::_Destroy().

max_size() [1/2]

```
static constexpr size_type std::allocator_traits< allocator< void > >::max_size (
            const allocator< void > & __a)  [inline], [static], [constexpr], [noexcept]
```

The maximum supported allocation size.

Parameters

_↩ _a	An allocator.

Returns

__a.max_size() or numeric_limits<size_type>::max()

Returns __a.max_size() if that expression is well-formed, otherwise returns numeric_limits<size_↩ type>::max()

max_size() [2/2]

```
static size_type std::allocator_traits< allocator< void > >::max_size (
            const allocator_type & )  [static], [delete]
```

max_size is ill-formed for allocator<void>

select_on_container_copy_construction() [1/2]

```
static constexpr allocator< void > std::allocator_traits< allocator< void > >::select_on_↩
container_copy_construction (
            const allocator< void > & __rhs)  [inline], [static], [constexpr]
```

Obtain an allocator to use when copying a container.

Parameters

__rhs	An allocator.

Returns

__rhs.select_on_container_copy_construction() or *__rhs*

Returns __rhs.select_on_container_copy_construction() if that expression is well-formed, otherwise returns *__rhs*

select_on_container_copy_construction() [2/2]

```
static constexpr allocator_type std::allocator_traits< allocator< void > >::select_on_container↩
_copy_construction (
            const allocator_type & __rhs)  [inline], [static], [constexpr]
```

Obtain an allocator to use when copying a container.

Parameters

__rhs	An allocator.

Returns

__rhs

The documentation for this struct was generated from the following file:

- bits/alloc_traits.h

5.191 std::allocator_traits< pmr::polymorphic_allocator< _Tp > > Struct Template Reference

#include <memory_resource>

Public Types

- typedef pmr::polymorphic_allocator< _Tp > allocator_type
- using allocator_type
- using const_pointer
- using const_pointer
- using const_void_pointer
- using const_void_pointer
- using difference_type
- using difference_type
- using is_always_equal
- using is_always_equal
- using pointer
- using pointer
- using propagate_on_container_copy_assignment
- using propagate_on_container_move_assignment
- using propagate_on_container_swap
- using **rebind_alloc**
- template<typename _Up>
  using **rebind_alloc**
- using **rebind_traits**
- template<typename _Up>
  using **rebind_traits**
- using size_type
- using size_type
- typedef pmr::polymorphic_allocator< _Tp >::value_type value_type
- using value_type
- using void_pointer
- using void_pointer

Static Public Member Functions

- static pointer allocate (allocator_type &__a, size_type __n)
- static pointer allocate (allocator_type &__a, size_type __n, const_void_pointer)
- static constexpr pointer allocate (pmr::polymorphic_allocator< _Tp > &__a, size_type __n)
- static constexpr pointer allocate (pmr::polymorphic_allocator< _Tp > &__a, size_type __n, const_void_pointer __hint)
- template<typename _Up, typename... _Args>
  static void construct (allocator_type &__a, _Up *__p, _Args &&... __args)
- static constexpr void construct (pmr::polymorphic_allocator< _Tp > &__a, _Tp *__p, _Args &&... __args) noexcept(_S_nothrow_construct< _Tp, _Args... >())
- static void deallocate (allocator_type &__a, pointer __p, size_type __n)
- static constexpr void deallocate (pmr::polymorphic_allocator< _Tp > &__a, pointer __p, size_type __n)
- template<typename _Up>
  static constexpr void destroy (allocator_type &, _Up *__p) noexcept(is_nothrow_destructible< _Up >::value)
- static constexpr void destroy (pmr::polymorphic_allocator< _Tp > &__a, _Tp *__p) noexcept(_S_nothrow_↩ destroy< _Tp >())
- static constexpr size_type max_size (const allocator_type &) noexcept
- static constexpr size_type max_size (const pmr::polymorphic_allocator< _Tp > &__a) noexcept
- static constexpr pmr::polymorphic_allocator< _Tp > select_on_container_copy_construction (const pmr::polymorphic_allocator< _Tp > &__rhs)
- using propagate_on_container_copy_assignment
- using propagate_on_container_move_assignment
- using propagate_on_container_swap
- static allocator_type select_on_container_copy_construction (const allocator_type &) noexcept

5.191.1 Detailed Description

template<typename _Tp>
struct std::allocator_traits< pmr::polymorphic_allocator< _Tp > >

Partial specialization for `std::pmr::polymorphic_allocator`

Since

C++17

5.191.2 Member Typedef Documentation

allocator_type [1/2]

```
typedef pmr::polymorphic_allocator< _Tp > std::allocator_traits< pmr::polymorphic_allocator< _Tp
> >::allocator_type
```

The allocator type.

allocator_type [2/2]

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::allocator_type
```

The allocator type.

const_pointer [1/2]

```
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::const_pointer
```

The allocator's const pointer type.
`Alloc::const_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<const value_type>`

const_pointer [2/2]

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::const_pointer
```

The allocator's const pointer type.

const_void_pointer [1/2]

```
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::const_void_pointer
```

The allocator's const void pointer type.
`Alloc::const_void_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<const void>`

const_void_pointer [2/2]

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::const_void_pointer
```

The allocator's const void pointer type.

difference_type [1/2]

```
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::difference_type
```

The allocator's difference type.
`Alloc::difference_type` if that type exists, otherwise `pointer_traits<pointer>::difference_type`

difference_type [2/2]

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::difference_type
```

The allocator's difference type.

is_always_equal [1/2]

```
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::is_always_equal
```

Whether all instances of the allocator type compare equal.
`Alloc::is_always_equal` if that type exists, otherwise `is_empty<Alloc>::type`

is_always_equal [2/2]

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::is_always_equal
```

Whether all instances of the allocator type compare equal.

pointer [1/2]

```
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::pointer
```

The allocator's pointer type.
`Alloc::pointer` if that type exists, otherwise `value_type*`

pointer [2/2]

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::pointer
```

The allocator's pointer type.

propagate_on_container_copy_assignment [1/2]

```
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::propagate_on_container_copy_↩
assignment
```

How the allocator is propagated on copy assignment.
`Alloc::propagate_on_container_copy_assignment` if that type exists, otherwise `false_type`

propagate_on_container_copy_assignment [2/2]

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::propagate_on_container_copy_↩
assignment
```

A `polymorphic_allocator` does not propagate when a container is copied, moved, or swapped.

propagate_on_container_move_assignment [1/2]

```
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::propagate_on_container_move_↩
assignment
```

How the allocator is propagated on move assignment.
`Alloc::propagate_on_container_move_assignment` if that type exists, otherwise `false_type`

propagate_on_container_move_assignment [2/2]

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::propagate_on_container_move_↩
assignment
```

A `polymorphic_allocator` does not propagate when a container is copied, moved, or swapped.

propagate_on_container_swap [1/2]

```
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::propagate_on_container_swap
```

How the allocator is propagated on swap.
`Alloc::propagate_on_container_swap` if that type exists, otherwise `false_type`

propagate_on_container_swap [2/2]

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::propagate_on_container_swap
```

A `polymorphic_allocator` does not propagate when a container is copied, moved, or swapped.

size_type [1/2]

```
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::size_type
```

The allocator's size type.
`Alloc::size_type` if that type exists, otherwise `make_unsigned<difference_type>::type`

size_type [2/2]

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::size_type
```

The allocator's size type.

value_type [1/2]

```
typedef pmr::polymorphic_allocator< _Tp >::value_type std::allocator_traits< pmr::polymorphic_allocator<
_Tp > >::value_type
```

The allocated type.

value_type [2/2]

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::value_type
```

The allocated type.

void_pointer [1/2]

```
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::void_pointer
```

The allocator's void pointer type.
`Alloc::void_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<void>`

void_pointer [2/2]

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::void_pointer
```

The allocator's void pointer type.

5.191.3 Member Function Documentation

allocate() [1/4]

```
template<typename _Tp>
static pointer std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::allocate (
            allocator_type & __a,
            size_type __n)  [inline], [static], [nodiscard]
```

Allocate memory.

Parameters

_↩ _a	An allocator.
_↩ _n	The number of objects to allocate space for.

Calls `a.allocate(n)`.

allocate() [2/4]

```
template<typename _Tp>
static pointer std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::allocate (
            allocator_type & __a,
            size_type __n,
            const_void_pointer )  [inline], [static], [nodiscard]
```

Allocate memory.

Parameters

_↩ _a	An allocator.
_↩ _n	The number of objects to allocate space for.

Returns

Memory of suitable size and alignment for `n` objects of type `value_type`.

The third parameter is ignored..
Returns `a.allocate(n)`.

allocate() [3/4]

```
static constexpr pointer std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::allocate (
            pmr::polymorphic_allocator< _Tp > & __a,
            size_type __n)  [inline], [static], [nodiscard], [constexpr]
```

Allocate memory.

Parameters

_↩ _a	An allocator.
_↩ _n	The number of objects to allocate space for.

Calls `a.allocate(n)`

allocate() [4/4]

```
static constexpr pointer std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::allocate (
            pmr::polymorphic_allocator< _Tp > & __a,
            size_type __n,
            const_void_pointer __hint)  [inline], [static], [nodiscard], [constexpr]
```

Allocate memory.

Parameters

__a	An allocator.
__n	The number of objects to allocate space for.
__hint	Aid to locality.

Returns

Memory of suitable size and alignment for *n* objects of type `value_type`

Returns `a.allocate(n, hint)` if that expression is well-formed, otherwise returns `a.allocate(n)`

construct() [1/2]

```
template<typename _Tp>
template<typename _Up, typename... _Args>
static void std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::construct (
            allocator_type & __a,
            _Up * __p,
            _Args &&... __args)  [inline], [static]
```

Construct an object of type `_Up`

Parameters

__a	An allocator.

Parameters

__p	Pointer to memory of suitable size and alignment for an object of type _Up.
__args	Constructor arguments.

Calls `__a.construct(__p, std::forward<_Args>(__args)...)` in C++11, C++14 and C++17. Changed in C++20 to call `std::construct_at(__p, std::forward<_Args>(__args)...)` instead. References std::forward().

construct() [2/2]

```
static constexpr void std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::construct (
            pmr::polymorphic_allocator< _Tp > & __a,
            _Tp * __p,
            _Args &&... __args)  [inline], [static], [constexpr], [noexcept]
```

Construct an object of type _Tp

Parameters

__a	An allocator.
__p	Pointer to memory of suitable size and alignment for Tp
__args	Constructor arguments.

Calls `__a.construct(__p, std::forward<Args>(__args)...)` if that expression is well-formed, otherwise uses placement-new to construct an object of type *_Tp* at location *__p* from the arguments *__args*...

deallocate() [1/2]

```
template<typename _Tp>
static void std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::deallocate (
            allocator_type & __a,
            pointer __p,
            size_type __n)  [inline], [static]
```

Deallocate memory.

Parameters

_↩ _a	An allocator.
_↩ _p	Pointer to the memory to deallocate.
_↩ _n	The number of objects space was allocated for.

Calls `a.deallocate(p, n)`.

deallocate() [2/2]

```
static constexpr void std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::deallocate (
            pmr::polymorphic_allocator< _Tp > & __a,
            pointer __p,
            size_type __n)  [inline], [static], [constexpr]
```

Deallocate memory.

Parameters

_↩ _a	An allocator.
_↩ _p	Pointer to the memory to deallocate.
_↩ _n	The number of objects space was allocated for.

Calls `a.deallocate(p, n)`

destroy() [1/2]

```
template<typename _Tp>
template<typename _Up>
static constexpr void std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::destroy (
            allocator_type & ,
            _Up * __p)  [inline], [static], [constexpr], [noexcept]
```

Destroy an object of type _Up

Parameters

_↩ _a	An allocator.
_↩ _p	Pointer to the object to destroy

Calls `p->_Up()`.

destroy() [2/2]

```
static constexpr void std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::destroy (
            pmr::polymorphic_allocator< _Tp > & __a,
            _Tp * __p)  [inline], [static], [constexpr], [noexcept]
```

Destroy an object of type *_Tp*.

Parameters

_↩ _a	An allocator.
_↩ _p	Pointer to the object to destroy

Calls `__a.destroy(__p)` if that expression is well-formed, otherwise calls `__p->~_Tp()`

max_size() [1/2]

```
template<typename _Tp>
static constexpr size_type std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::max_size
(
            const allocator_type & )  [inline], [static], [constexpr], [noexcept]
```

The maximum supported allocation size.

Returns

`numeric_limits<size_t>::max() / sizeof(value_type)`

max_size() [2/2]

```
static constexpr size_type std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::max_size
(
            const pmr::polymorphic_allocator< _Tp > & __a)  [inline], [static], [constexpr],
[noexcept]
```

The maximum supported allocation size.

Parameters

_↩ _a	An allocator.

Returns

`__a.max_size()` or `numeric_limits<size_type>::max()`

Returns `__a.max_size()` if that expression is well-formed, otherwise returns `numeric_limits<size_↩ type>::max()`

select_on_container_copy_construction() [1/2]

```
template<typename _Tp>
static allocator_type std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::select_on_↩
container_copy_construction (
            const allocator_type & )  [inline], [static], [noexcept]
```

A `polymorphic_allocator` does not propagate when a container is copied, moved, or swapped.

select_on_container_copy_construction() [2/2]

```
static constexpr pmr::polymorphic_allocator< _Tp > std::allocator_traits< pmr::polymorphic_allocator<
_Tp > >::select_on_container_copy_construction (
            const pmr::polymorphic_allocator< _Tp > & __rhs)  [inline], [static], [constexpr]
```

Obtain an allocator to use when copying a container.

Parameters

__rhs	An allocator.

Returns

`__rhs.select_on_container_copy_construction()` or *__rhs*

Returns `__rhs.select_on_container_copy_construction()` if that expression is well-formed, otherwise returns *__rhs*

The documentation for this struct was generated from the following file:

- memory_resource.h

5.192 __gnu_cxx::limit_condition::always_adjustor Struct Reference

`#include <throw_allocator.h>`

5.192.1 Detailed Description

Always enter the condition.
The documentation for this struct was generated from the following file:

- throw_allocator.h

5.193 __gnu_cxx::random_condition::always_adjustor Struct Reference

`#include <throw_allocator.h>`

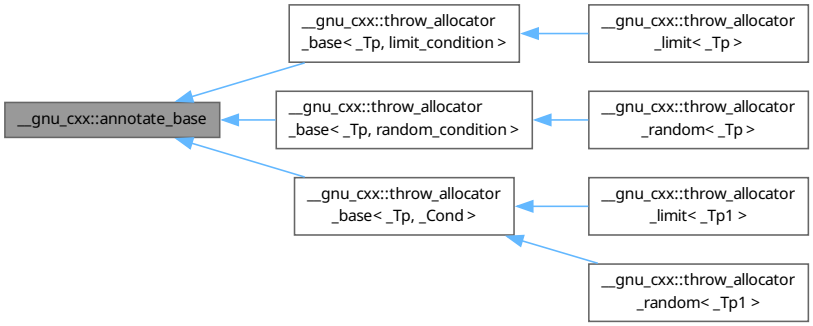
5.193.1 Detailed Description

Always enter the condition.
The documentation for this struct was generated from the following file:

- throw_allocator.h

5.194 __gnu_cxx::annotate_base Struct Reference

`#include <throw_allocator.h>`
Inheritance diagram for __gnu_cxx::annotate_base:



Public Member Functions

- void **check** (size_t label)
- map_alloc_type::iterator **check_allocated** (void ∗p, size_t size)
- void **check_constructed** (size_t label)
- map_construct_type::iterator **check_constructed** (void ∗p)
- void **erase** (void ∗p, size_t size)
- void **erase_construct** (void ∗p)
- void **insert** (void ∗p, size_t size)
- void **insert_construct** (void ∗p)

Static Public Member Functions

- static void **check** ()
- static size_t **get_label** ()
- static void **set_label** (size_t l)

Friends

- std::ostream & **operator**<< (std::ostream &os, const annotate_base &__b)

5.194.1 Detailed Description

Base class for checking address and label information about allocations. Create a std::map between the allocated address (void*) and a datum for annotations, which are a pair of numbers corresponding to label and allocated size.
The documentation for this struct was generated from the following file:

- throw_allocator.h

5.195 std::experimental::fundamentals_v1::any Class Reference

`#include <any>`

Public Member Functions

- any () noexcept
- template<typename _ValueType, typename _Tp = _Decay<_ValueType>, typename _Mgr = _Manager<_Tp>, typename enable_if< is_constructible< _Tp, _ValueType && >::value, bool >::type = true>
  any (_ValueType &&__value)
- template<typename _ValueType, typename _Tp = _Decay<_ValueType>, typename _Mgr = _Manager<_Tp>, typename enable_if<!is_constructible< _Tp, _ValueType && >::value, bool >::type = false>
  any (_ValueType &&__value)
- any (any &&__other) noexcept
- any (const any &__other)
- ~any ()
- void clear () noexcept
- bool empty () const noexcept
- template<typename _ValueType>
  enable_if_t<!is_same< any, decay_t< _ValueType > >::value, any & > operator= (_ValueType &&__rhs)
- any & operator= (any &&__rhs) noexcept
- any & operator= (const any &__rhs)
- void swap (any &__rhs) noexcept
- const type_info & type () const noexcept

Static Public Member Functions

- template<typename _Tp>
  static constexpr bool **__is_valid_cast** ()

Friends

- template<typename _Tp>
  enable_if_t< is_object< _Tp >::value, void * > **__any_caster** (const any *__any)

5.195.1 Detailed Description

A type-safe container of any type.
An `any` object's state is either empty or it stores a contained object of CopyConstructible type.

5.195.2 Constructor & Destructor Documentation

any() [1/5]

```
std::experimental::fundamentals_v1::any::any ()  [inline], [noexcept]
```

Default constructor, creates an empty object.

any() [2/5]

```
std::experimental::fundamentals_v1::any::any (
            const any & __other)  [inline]
```

Copy constructor, copies the state of `__other`.

any() [3/5]

```
std::experimental::fundamentals_v1::any::any (
            any && __other)  [inline], [noexcept]
```

Move constructor, transfer the state from `__other`.

Postcondition

`__other.empty()` (this postcondition is a GNU extension)

any() [4/5]

```
template<typename _ValueType, typename _Tp = _Decay<_ValueType>, typename _Mgr = _Manager<_Tp>,
typename enable_if< is_constructible< _Tp, _ValueType && >::value, bool >::type = true>
std::experimental::fundamentals_v1::any::any (
            _ValueType && __value)  [inline]
```

Construct with a copy of `__value` as the contained object.

any() [5/5]

```
template<typename _ValueType, typename _Tp = _Decay<_ValueType>, typename _Mgr = _Manager<_Tp>,
typename enable_if<!is_constructible< _Tp, _ValueType && >::value, bool >::type = false>
std::experimental::fundamentals_v1::any::any (
            _ValueType && __value)  [inline]
```

Construct with a copy of `__value` as the contained object.

~any()

```
std::experimental::fundamentals_v1::any::~any ()  [inline]
```

Destructor, calls `clear()`

5.195.3 Member Function Documentation

clear()

```
void std::experimental::fundamentals_v1::any::clear ()  [inline], [noexcept]
```

If not empty, destroy the contained object.

empty()

```
bool std::experimental::fundamentals_v1::any::empty () const  [inline], [nodiscard], [noexcept]
```

Reports whether there is a contained object or not.

operator=() [1/3]

```
template<typename _ValueType>
enable_if_t<!is_same< any, decay_t< _ValueType > >::value, any & > std::experimental::fundamentals↩
_v1::any::operator= (
            _ValueType && __rhs)  [inline]
```

Store a copy of __rhs as the contained object.

operator=() [2/3]

```
any & std::experimental::fundamentals_v1::any::operator= (
            any && __rhs)  [inline], [noexcept]
```

Move assignment operator.

Postcondition

__rhs.empty() (not guaranteed for other implementations)

operator=() [3/3]

```
any & std::experimental::fundamentals_v1::any::operator= (
            const any & __rhs)  [inline]
```

Copy the state of another object.

swap()

```
void std::experimental::fundamentals_v1::any::swap (
            any & __rhs)  [inline], [noexcept]
```

Exchange state with another object.

type()

```
const type_info & std::experimental::fundamentals_v1::any::type () const  [inline], [noexcept]
```

The typeid of the contained object, or typeid(void) if empty.
The documentation for this class was generated from the following file:

- experimental/any

5.196 std::array< _Tp, _Nm > Struct Template Reference

`#include <array>`

Public Types

- typedef const value_type * **const_iterator**
- typedef const value_type * **const_pointer**
- typedef const value_type & **const_reference**
- typedef std::reverse_iterator< const_iterator > **const_reverse_iterator**
- typedef std::ptrdiff_t **difference_type**
- typedef value_type * **iterator**
- typedef value_type * **pointer**
- typedef value_type & **reference**
- typedef std::reverse_iterator< iterator > **reverse_iterator**
- typedef std::size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- constexpr reference **at** (size_type __n)
- constexpr const_reference **at** (size_type __n) const
- constexpr const_reference **back** () const noexcept
- constexpr reference **back** () noexcept
- constexpr const_iterator **begin** () const noexcept
- constexpr iterator **begin** () noexcept
- constexpr const_iterator **cbegin** () const noexcept
- constexpr const_iterator **cend** () const noexcept
- constexpr const_reverse_iterator **crbegin** () const noexcept
- constexpr const_reverse_iterator **crend** () const noexcept
- constexpr const_pointer **data** () const noexcept
- constexpr pointer **data** () noexcept
- constexpr bool **empty** () const noexcept
- constexpr const_iterator **end** () const noexcept
- constexpr iterator **end** () noexcept
- constexpr void **fill** (const value_type &__u)
- constexpr const_reference **front** () const noexcept
- constexpr reference **front** () noexcept
- constexpr size_type **max_size** () const noexcept
- constexpr const_reference **operator[ ]** (size_type __n) const noexcept
- constexpr reference **operator[ ]** (size_type __n) noexcept
- constexpr const_reverse_iterator **rbegin** () const noexcept
- constexpr reverse_iterator **rbegin** () noexcept
- constexpr const_reverse_iterator **rend** () const noexcept
- constexpr reverse_iterator **rend** () noexcept
- constexpr size_type **size** () const noexcept
- constexpr void **swap** (array &__other) noexcept(__array_traits< _Tp, _Nm >::_Is_nothrow_swappable::value)

Public Attributes

- __array_traits< _Tp, _Nm >::_Type **_M_elems**

5.196.1 Detailed Description

template<typename _Tp, std::size_t _Nm>
struct std::array< _Tp, _Nm >

A standard container for storing a fixed size sequence of elements.
Meets the requirements of a `container`, a `reversible container`, and a `sequence`.
Sets support random access iterators.

Template Parameters

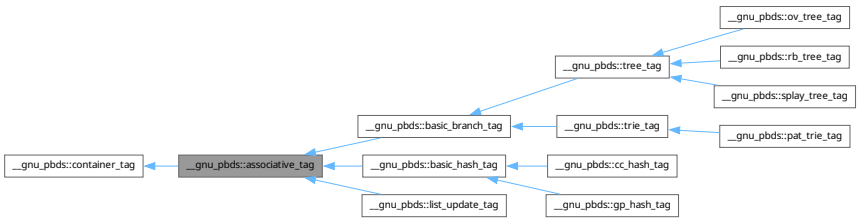
Tp	Type of element. Required to be a complete type.
Nm	Number of elements.

The documentation for this struct was generated from the following file:

- array

5.197 __gnu_pbds::associative_tag Struct Reference

`#include <tag_and_trait.hpp>`
Inheritance diagram for __gnu_pbds::associative_tag:



5.197.1 Detailed Description

Basic associative-container.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

5.198 std::atomic< _Tp > Struct Template Reference

`#include <atomic>`

Public Types

- using **value_type**

Public Member Functions

- constexpr **atomic** (_Tp __i) noexcept
- **atomic** (const atomic &)=delete
- bool **compare_exchange_strong** (_Tp &__e, _Tp __i, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_strong** (_Tp &__e, _Tp __i, memory_order __m=memory_order_seq_cst) volatile noexcept
- bool **compare_exchange_strong** (_Tp &__e, _Tp __i, memory_order __s, memory_order __f) noexcept
- bool **compare_exchange_strong** (_Tp &__e, _Tp __i, memory_order __s, memory_order __f) volatile noexcept
- bool **compare_exchange_weak** (_Tp &__e, _Tp __i, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_weak** (_Tp &__e, _Tp __i, memory_order __m=memory_order_seq_cst) volatile noexcept
- bool **compare_exchange_weak** (_Tp &__e, _Tp __i, memory_order __s, memory_order __f) noexcept
- bool **compare_exchange_weak** (_Tp &__e, _Tp __i, memory_order __s, memory_order __f) volatile noexcept
- _Tp **exchange** (_Tp __i, memory_order __m=memory_order_seq_cst) noexcept
- _Tp **exchange** (_Tp __i, memory_order __m=memory_order_seq_cst) volatile noexcept
- bool **is_lock_free** () const noexcept
- bool **is_lock_free** () const volatile noexcept
- _Tp **load** (memory_order __m=memory_order_seq_cst) const noexcept
- _Tp **load** (memory_order __m=memory_order_seq_cst) const volatile noexcept
- **operator _Tp** () const noexcept
- **operator _Tp** () const volatile noexcept
- _Tp **operator=** (_Tp __i) noexcept

- _Tp **operator=** (_Tp __i) volatile noexcept
- atomic & **operator=** (const atomic &) volatile =delete
- atomic & **operator=** (const atomic &)=delete
- void **store** (_Tp __i, memory_order __m=memory_order_seq_cst) noexcept
- void **store** (_Tp __i, memory_order __m=memory_order_seq_cst) volatile noexcept

5.198.1 Detailed Description

template<typename _Tp>
struct std::atomic< _Tp >

Generic atomic type, primary class template.

Template Parameters

_Tp	Type to be made atomic, must be trivially copyable.

The documentation for this struct was generated from the following file:

- atomic

5.199 std::atomic< _Tp * > Struct Template Reference

`#include <atomic>`

Public Types

- typedef __atomic_base< _Tp * > **__base_type**
- typedef _Tp * **__pointer_type**
- using **difference_type**
- using **value_type**
- using **value_type**

Public Member Functions

- constexpr **atomic** (__pointer_type __p) noexcept
- constexpr **atomic** (_Tp __i) noexcept
- **atomic** (const atomic &)=delete
- **atomic** (const atomic &)=delete
- bool **compare_exchange_strong** (__pointer_type &__p1, __pointer_type __p2, memory_order __m1, memory_order __m2) noexcept
- bool **compare_exchange_strong** (__pointer_type &__p1, __pointer_type __p2, memory_order __m1, memory_order __m2) volatile noexcept
- bool **compare_exchange_strong** (__pointer_type &__p1, __pointer_type __p2, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_strong** (__pointer_type &__p1, __pointer_type __p2, memory_order __m=memory_order_seq_cst) volatile noexcept
- bool **compare_exchange_strong** (_Tp &__e, _Tp __i, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_strong** (_Tp &__e, _Tp __i, memory_order __m=memory_order_seq_cst) volatile noexcept
- bool **compare_exchange_strong** (_Tp &__e, _Tp __i, memory_order __s, memory_order __f) noexcept
- bool **compare_exchange_strong** (_Tp &__e, _Tp __i, memory_order __s, memory_order __f) volatile noexcept
- bool **compare_exchange_weak** (__pointer_type &__p1, __pointer_type __p2, memory_order __m1, memory_order __m2) noexcept

- bool **compare_exchange_weak** (__pointer_type &__p1, __pointer_type __p2, memory_order __m1, memory_order __m2) volatile noexcept
- bool **compare_exchange_weak** (__pointer_type &__p1, __pointer_type __p2, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_weak** (__pointer_type &__p1, __pointer_type __p2, memory_order __m=memory_order_seq_cst) volatile noexcept
- bool **compare_exchange_weak** (_Tp &__e, _Tp __i, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_weak** (_Tp &__e, _Tp __i, memory_order __m=memory_order_seq_cst) volatile noexcept
- bool **compare_exchange_weak** (_Tp &__e, _Tp __i, memory_order __s, memory_order __f) noexcept
- bool **compare_exchange_weak** (_Tp &__e, _Tp __i, memory_order __s, memory_order __f) volatile noexcept
- __pointer_type **exchange** (__pointer_type __p, memory_order __m=memory_order_seq_cst) noexcept
- __pointer_type **exchange** (__pointer_type __p, memory_order __m=memory_order_seq_cst) volatile noexcept
- _Tp **exchange** (_Tp __i, memory_order __m=memory_order_seq_cst) noexcept
- _Tp **exchange** (_Tp __i, memory_order __m=memory_order_seq_cst) volatile noexcept
- __pointer_type **fetch_add** (ptrdiff_t __d, memory_order __m=memory_order_seq_cst) noexcept
- __pointer_type **fetch_add** (ptrdiff_t __d, memory_order __m=memory_order_seq_cst) volatile noexcept
- __pointer_type **fetch_sub** (ptrdiff_t __d, memory_order __m=memory_order_seq_cst) noexcept
- __pointer_type **fetch_sub** (ptrdiff_t __d, memory_order __m=memory_order_seq_cst) volatile noexcept
- bool **is_lock_free** () const noexcept
- bool **is_lock_free** () const noexcept
- bool **is_lock_free** () const volatile noexcept
- bool **is_lock_free** () const volatile noexcept
- _Tp **load** (memory_order __m=memory_order_seq_cst) const noexcept
- __pointer_type **load** (memory_order __m=memory_order_seq_cst) const noexcept
- _Tp **load** (memory_order __m=memory_order_seq_cst) const volatile noexcept
- __pointer_type **load** (memory_order __m=memory_order_seq_cst) const volatile noexcept
- **operator __pointer_type** () const noexcept
- **operator __pointer_type** () const volatile noexcept
- **operator _Tp** () const noexcept
- **operator _Tp** () const volatile noexcept
- __pointer_type **operator++** () noexcept
- __pointer_type **operator++** () volatile noexcept
- __pointer_type **operator++** (int) noexcept
- __pointer_type **operator++** (int) volatile noexcept
- __pointer_type **operator+=** (ptrdiff_t __d) noexcept
- __pointer_type **operator+=** (ptrdiff_t __d) volatile noexcept
- __pointer_type **operator--** () noexcept
- __pointer_type **operator--** () volatile noexcept
- __pointer_type **operator--** (int) noexcept
- __pointer_type **operator--** (int) volatile noexcept
- __pointer_type **operator-=** (ptrdiff_t __d) noexcept
- __pointer_type **operator-=** (ptrdiff_t __d) volatile noexcept
- __pointer_type **operator=** (__pointer_type __p) noexcept
- __pointer_type **operator=** (__pointer_type __p) volatile noexcept
- _Tp **operator=** (_Tp __i) noexcept
- _Tp **operator=** (_Tp __i) volatile noexcept
- atomic & **operator=** (const atomic &) volatile =delete
- atomic & **operator=** (const atomic &) volatile=delete
- atomic & **operator=** (const atomic &)=delete
- atomic & **operator=** (const atomic &)=delete

- void **store** (__pointer_type __p, memory_order __m=memory_order_seq_cst) noexcept
- void **store** (__pointer_type __p, memory_order __m=memory_order_seq_cst) volatile noexcept
- void **store** (_Tp __i, memory_order __m=memory_order_seq_cst) noexcept
- void **store** (_Tp __i, memory_order __m=memory_order_seq_cst) volatile noexcept

Public Attributes

- __base_type **_M_b**

5.199.1 Detailed Description

template<**typename _Tp**>
struct std::atomic< **_Tp** * >

Partial specialization for pointer types.
The documentation for this struct was generated from the following file:

- atomic

5.200 std::atomic_flag Struct Reference

`#include <atomic_base.h>`

Public Member Functions

- constexpr **atomic_flag** (bool __i) noexcept
- **atomic_flag** (const atomic_flag &)=delete
- void **clear** (memory_order __m=memory_order_seq_cst) noexcept
- void **clear** (memory_order __m=memory_order_seq_cst) volatile noexcept
- atomic_flag & **operator=** (const atomic_flag &) volatile =delete
- atomic_flag & **operator=** (const atomic_flag &)=delete
- bool **test** (memory_order __m=memory_order_seq_cst) const noexcept
- bool **test** (memory_order __m=memory_order_seq_cst) const volatile noexcept
- bool **test_and_set** (memory_order __m=memory_order_seq_cst) noexcept
- bool **test_and_set** (memory_order __m=memory_order_seq_cst) volatile noexcept

5.200.1 Detailed Description

atomic_flag
The documentation for this struct was generated from the following file:

- atomic_base.h

5.201 std::auto_ptr< _Tp > Class Template Reference

`#include <memory>`

Public Types

- typedef _Tp element_type

Public Member Functions

- auto_ptr (auto_ptr &__a) throw ()
- template<typename _Tp1>
  auto_ptr (auto_ptr< _Tp1 > &__a) throw ()
- auto_ptr (auto_ptr_ref< element_type > __ref) throw ()
- auto_ptr (element_type *__p=0) throw ()
- ~auto_ptr ()
- element_type * get () const throw ()
- template<typename _Tp1>
  operator auto_ptr< _Tp1 > () throw ()
- template<typename _Tp1>
  operator auto_ptr_ref< _Tp1 > () throw ()
- element_type & operator* () const throw ()
- element_type * operator-> () const throw ()
- auto_ptr & operator= (auto_ptr &__a) throw ()
- template<typename _Tp1>
  auto_ptr & operator= (auto_ptr< _Tp1 > &__a) throw ()
- auto_ptr & **operator=** (auto_ptr_ref< element_type > __ref) throw ()
- element_type * release () throw ()
- void reset (element_type *__p=0) throw ()

5.201.1 Detailed Description

template<typename _Tp>
class std::auto_ptr< _Tp >

A simple smart pointer providing strict ownership semantics.
The Standard says:

```
An  auto_ptr owns the object it holds a pointer to.  Copying
an  auto_ptr copies the pointer and transfers ownership to the
destination.  If more than one  auto_ptr owns the same object
at the same time the behavior of the program is undefined.

The uses of  auto_ptr include providing temporary
exception-safety for dynamically allocated memory, passing
ownership of dynamically allocated memory to a function, and
returning dynamically allocated memory from a function.   auto_ptr does not meet the CopyConstructib
requirements for Standard Library  container elements and thus
instantiating a Standard Library container with an  auto_ptr results in undefined behavior.
```

Quoted from [20.4.5]/3.
Good examples of what can and cannot be done with auto_ptr can be found in the libstdc++ testsuite.
_GLIBCXX_RESOLVE_LIB_DEFECTS

1. auto_ptr<> conversion issues These resolutions have all been incorporated.

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

5.201.2 Member Typedef Documentation

element_type

```
template<typename _Tp>
typedef _Tp std::auto_ptr< _Tp >::element_type
```

The pointed-to type.

5.201.3 Constructor & Destructor Documentation

auto_ptr() [1/4]

```
template<typename _Tp>
std::auto_ptr< _Tp >::auto_ptr (
            element_type * __p = 0) throw ( )   [inline], [explicit]
```

An auto_ptr is usually constructed from a raw pointer.

Parameters

_↩ _p	A pointer (defaults to NULL).

This object now *owns* the object pointed to by *__p*.
Referenced by auto_ptr(), auto_ptr(), operator=(), and operator=().

auto_ptr() [2/4]

```
template<typename _Tp>
std::auto_ptr< _Tp >::auto_ptr (
            auto_ptr< _Tp > & __a) throw ( )   [inline]
```

An auto_ptr can be constructed from another auto_ptr.

Parameters

_↩ _a	Another auto_ptr of the same type.

This object now *owns* the object previously owned by *__a*, which has given up ownership.
References auto_ptr().

auto_ptr() [3/4]

```
template<typename _Tp>
template<typename _Tp1>
std::auto_ptr< _Tp >::auto_ptr (
            auto_ptr< _Tp1 > & __a) throw ( )   [inline]
```

An auto_ptr can be constructed from another auto_ptr.

Parameters

_↩ _a	Another auto_ptr of a different but related type.

A pointer-to-Tp1 must be convertible to a pointer-to-Tp/element_type.
This object now *owns* the object previously owned by *__a*, which has given up ownership.
References auto_ptr().

~auto_ptr()

```
template<typename _Tp>
std::auto_ptr< _Tp >::~auto_ptr ()  [inline]
```

When the auto_ptr goes out of scope, the object it owns is deleted. If it no longer owns anything (i.e., `get()` is `NULL`), then this has no effect.
The C++ standard says there is supposed to be an empty throw specification here, but omitting it is standard conforming. Its presence can be detected only if _Tp::~_Tp() throws, but this is prohibited. [17.4.3.6]/2

auto_ptr() [4/4]

```
template<typename _Tp>
std::auto_ptr< _Tp >::auto_ptr (
            auto_ptr_ref< element_type > __ref) throw ( )   [inline]
```

Automatic conversions.
These operations are supposed to convert an auto_ptr into and from an auto_ptr_ref automatically as needed. This would allow constructs such as

```
auto_ptr<Derived>  func_returning_auto_ptr(.....);
...
auto_ptr<Base> ptr = func_returning_auto_ptr(.....);
```

But it doesn't work, and won't be fixed. For further details see https://cplusplus.github.io/LWG/lwg-closed.html#463

5.201.4 Member Function Documentation

get()

```
template<typename _Tp>
element_type * std::auto_ptr< _Tp >::get () const throw ( )   [inline]
```

Bypassing the smart pointer.

Returns

The raw pointer being managed.

You can get a copy of the pointer that this object owns, for situations such as passing to a function which only accepts a raw pointer.

Note

This auto_ptr still owns the memory.

operator*()

```
template<typename _Tp>
element_type & std::auto_ptr< _Tp >::operator* () const throw ( )   [inline]
```

Smart pointer dereferencing.
If this auto_ptr no longer owns anything, then this operation will crash. (For a smart pointer, *no longer owns anything* is the same as being a null pointer, and you know what happens when you dereference one of those...)

operator->()

```
template<typename _Tp>
element_type * std::auto_ptr< _Tp >::operator-> () const throw ( )   [inline]
```

Smart pointer dereferencing.
This returns the pointer itself, which the language then will automatically cause to be dereferenced.

operator=() [1/2]

```
template<typename _Tp>
auto_ptr & std::auto_ptr< _Tp >::operator= (
            auto_ptr< _Tp > & __a) throw ( )   [inline]
```

auto_ptr assignment operator.

Parameters

_↩ _a	Another auto_ptr of the same type.

This object now *owns* the object previously owned by __a, which has given up ownership. The object that this one *used* to own and track has been deleted.
References auto_ptr(), and reset().

operator=() [2/2]

```
template<typename _Tp>
template<typename _Tp1>
auto_ptr & std::auto_ptr< _Tp >::operator= (
            auto_ptr< _Tp1 > & __a) throw ( )   [inline]
```

auto_ptr assignment operator.

Parameters

_↩ _a	Another auto_ptr of a different but related type.

A pointer-to-Tp1 must be convertible to a pointer-to-Tp/element_type.
This object now *owns* the object previously owned by __a, which has given up ownership. The object that this one *used* to own and track has been deleted.
References auto_ptr(), and reset().

release()

```
template<typename _Tp>
element_type * std::auto_ptr< _Tp >::release () throw ( )   [inline]
```

Bypassing the smart pointer.

Returns

The raw pointer being managed.

You can get a copy of the pointer that this object owns, for situations such as passing to a function which only accepts a raw pointer.

Note

This auto_ptr no longer owns the memory. When this object goes out of scope, nothing will happen.

reset()

```
template<typename _Tp>
void std::auto_ptr< _Tp >::reset (
            element_type * __p = 0) throw ( )   [inline]
```

Forcibly deletes the managed object.

Parameters

_↩ _p	A pointer (defaults to NULL).

This object now *owns* the object pointed to by __*p*. The previous object has been deleted.
Referenced by operator=(), and operator=().
The documentation for this class was generated from the following files:

- shared_ptr_base.h
- auto_ptr.h

5.202 std::auto_ptr_ref< _Tp1 > Struct Template Reference

`#include <auto_ptr.h>`

Public Member Functions

- **auto_ptr_ref** (_Tp1 *__p)

Public Attributes

- _Tp1 * **_M_ptr**

5.202.1 Detailed Description

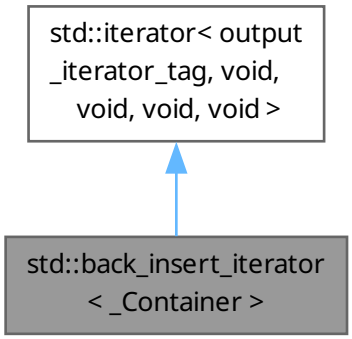
template<typename _Tp1>
struct std::auto_ptr_ref< _Tp1 >

A wrapper class to provide auto_ptr with reference semantics. For example, an auto_ptr can be assigned (or constructed from) the result of a function which returns an auto_ptr by value.
All the auto_ptr_ref stuff should happen behind the scenes.
The documentation for this struct was generated from the following file:

- auto_ptr.h

5.203 std::back_insert_iterator< _Container > Class Template Reference

`#include <stl_iterator.h>`
Inheritance diagram for std::back_insert_iterator< _Container >:



Public Types

- typedef _Container container_type
- using **difference_type**
- typedef output_iterator_tag iterator_category
- typedef void pointer
- typedef void reference
- typedef void value_type

Public Member Functions

- constexpr back_insert_iterator (_Container &__x)
- constexpr back_insert_iterator & operator* ()
- constexpr back_insert_iterator & operator++ ()
- constexpr back_insert_iterator operator++ (int)
- constexpr back_insert_iterator & operator= (const typename _Container::value_type &__value)
- constexpr back_insert_iterator & **operator=** (typename _Container::value_type &&__value)

Protected Attributes

- _Container * **container**

5.203.1 Detailed Description

template<typename _Container>
class std::back_insert_iterator< _Container >

Turns assignment into insertion.
These are output iterators, constructed from a container-of-T. Assigning a T to the iterator appends it to the container using push_back.
Tip: Using the back_inserter function to create these iterators can save typing.

5.203.2 Member Typedef Documentation

container_type

```
template<typename _Container>
typedef _Container std::back_insert_iterator< _Container >::container_type
```

A nested typedef for the type of whatever container you used.

iterator_category

```
typedef output_iterator_tag std::iterator< output_iterator_tag, void, void, void, void >::iterator↩
_category [inherited]
```

One of the tag types.

pointer

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::pointer [inherited]
```

This type represents a pointer-to-value_type.

reference

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::reference [inherited]
```

This type represents a reference-to-value_type.

value_type

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::value_type  [inherited]
```
The type "pointed to" by the iterator.

5.203.3 Constructor & Destructor Documentation

back_insert_iterator()

```
template<typename _Container>
std::back_insert_iterator< _Container >::back_insert_iterator (
            _Container & __x)  [inline], [explicit], [constexpr]
```
The only way to create this iterator is with a container.
References std::__addressof().

5.203.4 Member Function Documentation

operator∗()

```
template<typename _Container>
back_insert_iterator & std::back_insert_iterator< _Container >::operator* ()  [inline], [nodiscard],
[constexpr]
```
Simply returns ∗this.

operator++() [1/2]

```
template<typename _Container>
back_insert_iterator & std::back_insert_iterator< _Container >::operator++ ()  [inline], [constexpr]
```
Simply returns ∗this. (This iterator does not *move*.)

operator++() [2/2]

```
template<typename _Container>
back_insert_iterator std::back_insert_iterator< _Container >::operator++ (
            int )  [inline], [constexpr]
```
Simply returns ∗this. (This iterator does not *move*.)

operator=()

```
template<typename _Container>
back_insert_iterator & std::back_insert_iterator< _Container >::operator= (
            const typename _Container::value_type & __value)  [inline], [constexpr]
```

Parameters

__value	An instance of whatever type container_type::const_reference is; presumably a reference-to-const T for container<T>.

Returns

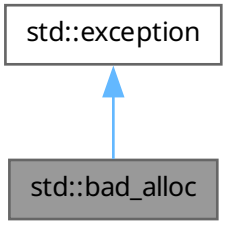
This iterator, for chained operations.

This kind of iterator doesn't really have a *position* in the container (you can think of the position as being permanently at the end, if you like). Assigning a value to the iterator will always append the value to the end of the container.
The documentation for this class was generated from the following file:

- bits/stl_iterator.h

5.204 std::bad_alloc Class Reference

`#include <new>`
Inheritance diagram for std::bad_alloc:



Public Member Functions

- **bad_alloc** (const bad_alloc &)=default
- bad_alloc & **operator=** (const bad_alloc &)=default
- virtual const char * what () const throw ()

5.204.1 Detailed Description

Exception possibly thrown by `new`.
`bad_alloc` (or classes derived from it) is used to report allocation errors from the throwing forms of `new`.

5.204.2 Member Function Documentation

what()

```
virtual const char * std::bad_alloc::what () const throw ( )   [virtual]
```

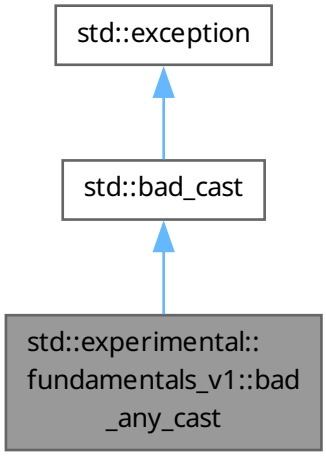
Returns a C-style character string describing the general cause of the current error.
Reimplemented from std::exception.
The documentation for this class was generated from the following file:

- new

5.205 std::experimental::fundamentals_v1::bad_any_cast Class Reference

`#include <any>`

Inheritance diagram for std::experimental::fundamentals_v1::bad_any_cast:



Public Member Functions

- virtual const char ∗ what () const noexcept

5.205.1 Detailed Description

Exception class thrown by a failed `any_cast`.

5.205.2 Member Function Documentation

what()

```
virtual const char * std::experimental::fundamentals_v1::bad_any_cast::what () const  [inline],
[virtual], [noexcept]
```

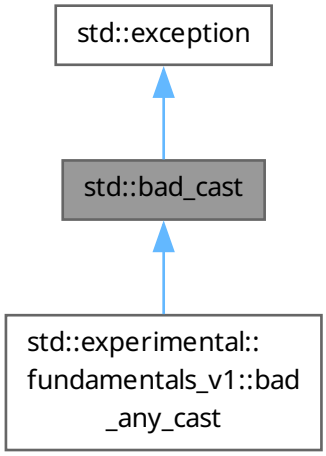
Returns a C-style character string describing the general cause of the current error.
Reimplemented from std::bad_cast.
The documentation for this class was generated from the following file:

- experimental/any

5.206 std::bad_cast Class Reference

```
#include <typeinfo>
```

Inheritance diagram for std::bad_cast:



Public Member Functions

- virtual const char ∗ what () const noexcept

5.206.1 Detailed Description

Thrown during incorrect typecasting.
If you attempt an invalid `dynamic_cast` expression, an instance of this class (or something derived from this class) is thrown.

5.206.2 Member Function Documentation

what()

```
virtual const char * std::bad_cast::what () const  [virtual], [noexcept]
```

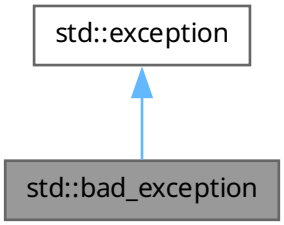
Returns a C-style character string describing the general cause of the current error.
Reimplemented from std::exception.
Reimplemented in std::experimental::fundamentals_v1::bad_any_cast.
The documentation for this class was generated from the following file:

- typeinfo

5.207 std::bad_exception Class Reference

```
#include <exception>
```

Inheritance diagram for std::bad_exception:



Public Member Functions

- virtual const char ∗ what () const noexcept

5.207.1 Detailed Description

If an exception is thrown which is not listed in a function's exception specification, one of these may be thrown.

5.207.2 Member Function Documentation

what()

```
virtual const char * std::bad_exception::what () const  [virtual], [noexcept]
```

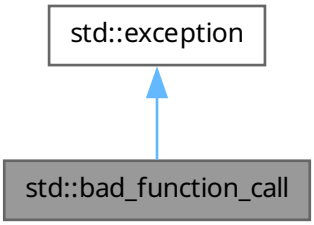
Returns a C-style character string describing the general cause of the current error.
Reimplemented from std::exception.
The documentation for this class was generated from the following file:

- exception

5.208 std::bad_function_call Class Reference

```
#include <std_function.h>
```

Inheritance diagram for std::bad_function_call:



Public Member Functions

- const char * what () const noexcept

5.208.1 Detailed Description

Exception class thrown when class template function's operator() is called with an empty target.

5.208.2 Member Function Documentation

what()

```
const char * std::bad_function_call::what () const  [virtual], [noexcept]
```

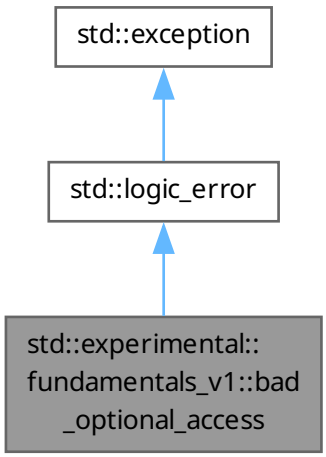
Returns a C-style character string describing the general cause of the current error.
Reimplemented from std::exception.
The documentation for this class was generated from the following file:

- std_function.h

5.209 std::experimental::fundamentals_v1::bad_optional_access Class Reference

```
#include <optional>
```

Inheritance diagram for std::experimental::fundamentals_v1::bad_optional_access:



Public Member Functions

- **bad_optional_access** (const char *__arg)
- virtual const char * what () const noexcept

5.209.1 Detailed Description

Exception class thrown when a disengaged optional object is dereferenced.

5.209.2 Member Function Documentation

what()

```
virtual const char * std::logic_error::what () const  [virtual], [noexcept], [inherited]
```

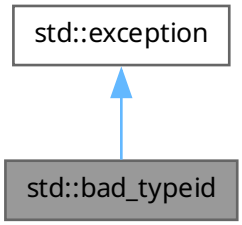
Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).
Reimplemented from std::exception.
Reimplemented in std::future_error.
The documentation for this class was generated from the following file:

- experimental/optional

5.210 std::bad_typeid Class Reference

```
#include <typeinfo>
```

Inheritance diagram for std::bad_typeid:



Public Member Functions

- virtual const char * what () const noexcept

5.210.1 Detailed Description

Thrown when a NULL pointer in a `typeid` expression is used.

5.210.2 Member Function Documentation

what()

```
virtual const char * std::bad_typeid::what () const  [virtual], [noexcept]
```

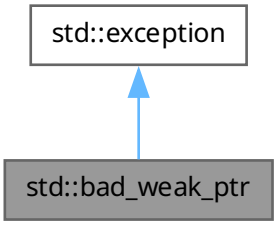
Returns a C-style character string describing the general cause of the current error.
Reimplemented from std::exception.
The documentation for this class was generated from the following file:

- typeinfo

5.211 std::bad_weak_ptr Class Reference

```
#include <shared_ptr_base.h>
```

Inheritance diagram for std::bad_weak_ptr:



Public Member Functions

- virtual char const * what () const noexcept

5.211.1 Detailed Description

Exception possibly thrown by `shared_ptr`.

5.211.2 Member Function Documentation

what()

```
virtual char const * std::bad_weak_ptr::what () const  [virtual], [noexcept]
```

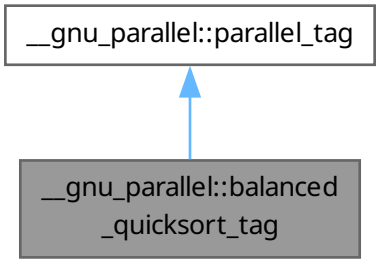
Returns a C-style character string describing the general cause of the current error.
Reimplemented from std::exception.
The documentation for this class was generated from the following file:

- shared_ptr_base.h

5.212 __gnu_parallel::balanced_quicksort_tag Struct Reference

```
#include <tags.h>
```

Inheritance diagram for __gnu_parallel::balanced_quicksort_tag:



Public Member Functions

- **balanced_quicksort_tag** (_ThreadIndex __num_threads)
- _ThreadIndex __get_num_threads ()
- void set_num_threads (_ThreadIndex __num_threads)

5.212.1 Detailed Description

Forces parallel sorting using balanced quicksort at compile time.

5.212.2 Member Function Documentation

__get_num_threads()

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ()  [inline], [inherited]
```

Find out desired number of threads.

Returns

Desired number of threads.

Referenced by __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), and __gnu_parallel::__parallel_sort(

set_num_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
            _ThreadIndex __num_threads)  [inline], [inherited]
```

Set the desired number of threads.

Parameters

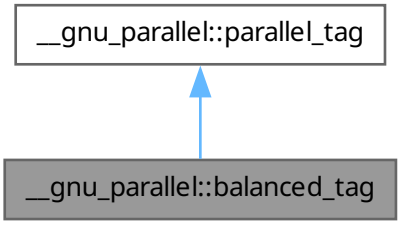
__num_threads	Desired number of threads.

The documentation for this struct was generated from the following file:

- tags.h

5.213 __gnu_parallel::balanced_tag Struct Reference

`#include <tags.h>`
Inheritance diagram for __gnu_parallel::balanced_tag:



Public Member Functions

- _ThreadIndex __get_num_threads ()
- void set_num_threads (_ThreadIndex __num_threads)

5.213.1 Detailed Description

Recommends parallel execution using dynamic load-balancing at compile time.

5.213.2 Member Function Documentation

__get_num_threads()

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ()  [inline], [inherited]
```
Find out desired number of threads.

Returns

Desired number of threads.

Referenced by __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), and __gnu_parallel::__parallel_sort(

set_num_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
            _ThreadIndex __num_threads)  [inline], [inherited]
```
Set the desired number of threads.

Parameters

__num_threads	Desired number of threads.

The documentation for this struct was generated from the following file:

- tags.h

5.214 std::tr2::bases< _Tp > Struct Template Reference

`#include <type_traits>`

Public Types

- typedef __reflection_typelist< __bases(_Tp)... > **type**

5.214.1 Detailed Description

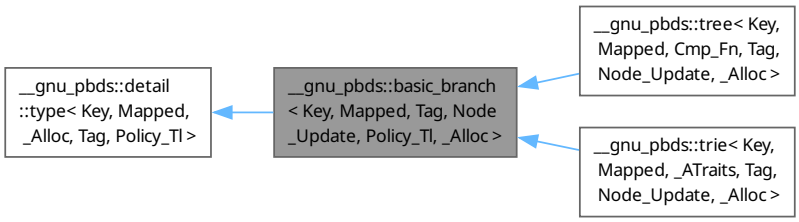
template<typename _Tp>
struct std::tr2::bases< _Tp >

Sequence abstraction metafunctions for manipulating a typelist.
Enumerate all the base classes of a class. Form of a typelist.
The documentation for this struct was generated from the following file:

- tr2/type_traits

5.215 __gnu_pbds::basic_branch< Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc > Class Template Reference

`#include <assoc_container.hpp>`
Inheritance diagram for __gnu_pbds::basic_branch< Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc >:



Public Types

- typedef Node_Update **node_update**

Protected Member Functions

- **basic_branch** (const basic_branch &other)
- template<typename T0>
  basic_branch (T0 t0)
- template<typename T0, typename T1>
  basic_branch (T0 t0, T1 t1)
- template<typename T0, typename T1, typename T2>
  basic_branch (T0 t0, T1 t1, T2 t2)
- template<typename T0, typename T1, typename T2, typename T3>
  basic_branch (T0 t0, T1 t1, T2 t2, T3 t3)
- template<typename T0, typename T1, typename T2, typename T3, typename T4>
  basic_branch (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4)

- template<typename T0, typename T1, typename T2, typename T3, typename T4, typename T5>
  basic_branch (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5)
- template<typename T0, typename T1, typename T2, typename T3, typename T4, typename T5, typename T6>
  basic_branch (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6)

5.215.1 Detailed Description

template<typename Key, typename Mapped, typename Tag, typename Node_Update, typename Policy_Tl, typename _Alloc>
class __gnu_pbds::basic_branch< Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc >

A branched, tree-like (tree, trie) container abstraction.

Template Parameters

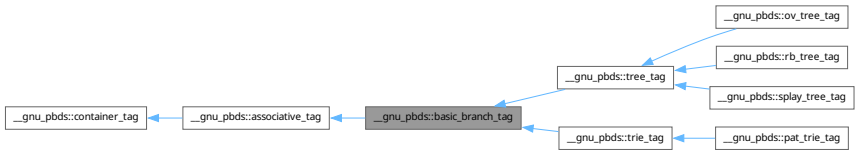
Key	Key type.
Mapped	Map type.
Tag	Instantiating data structure type, see container_tag.
Node_Update	Updates nodes, restores invariants.
Policy_TL	Policy typelist.
_Alloc	Allocator type.

Base is dispatched at compile time via Tag, from the following choices: tree_tag, trie_tag, and their descendants. Base choices are: detail::ov_tree_map, detail::rb_tree_map, detail::splay_tree_map, and detail::pat_trie_map.
The documentation for this class was generated from the following file:

- assoc_container.hpp

5.216 __gnu_pbds::basic_branch_tag Struct Reference

`#include <tag_and_trait.hpp>`
Inheritance diagram for __gnu_pbds::basic_branch_tag:



5.216.1 Detailed Description

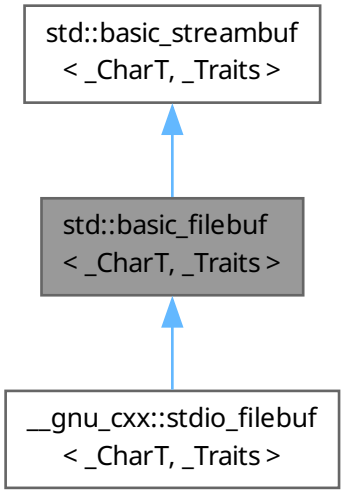
Basic branch structure.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

5.217 std::basic_filebuf< _CharT, _Traits > Class Template Reference

`#include <fstream>`

Inheritance diagram for std::basic_filebuf< _CharT, _Traits >:



Public Types

- typedef codecvt< char_type, char, __state_type > **__codecvt_type**
- typedef __basic_file< char > **__file_type**
- typedef basic_filebuf< char_type, traits_type > **__filebuf_type**
- typedef traits_type::state_type **__state_type**
- typedef basic_streambuf< char_type, traits_type > **__streambuf_type**
- typedef _CharT **char_type**
- typedef traits_type::int_type **int_type**
- typedef traits_type::off_type **off_type**
- typedef traits_type::pos_type **pos_type**
- typedef _Traits **traits_type**

Public Member Functions

- basic_filebuf ()
- **basic_filebuf** (basic_filebuf &&)
- **basic_filebuf** (const basic_filebuf &)=delete
- virtual ~basic_filebuf ()
- __filebuf_type * close ()
- locale getloc () const
- streamsize in_avail ()
- bool is_open () const throw ()
- template<typename _Path>
  _If_fs_path< _Path, __filebuf_type * > open (const _Path &__s, ios_base::openmode __mode)
- __filebuf_type * open (const char *__s, ios_base::openmode __mode)
- __filebuf_type * open (const std::string &__s, ios_base::openmode __mode)

- basic_filebuf & **operator=** (basic_filebuf &&)
- basic_filebuf & **operator=** (const basic_filebuf &)=delete
- locale pubimbue (const locale &__loc)
- int_type sbumpc ()
- int_type sgetc ()
- streamsize sgetn (char_type *__s, streamsize __n)
- int_type snextc ()
- int_type sputbackc (char_type __c)
- int_type sputc (char_type __c)
- streamsize sputn (const char_type *__s, streamsize __n)
- int_type sungetc ()
- void **swap** (basic_filebuf &)

- basic_streambuf * pubsetbuf (char_type *__s, streamsize __n)
- pos_type pubseekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)
- pos_type pubseekpos (pos_type __sp, ios_base::openmode __mode=ios_base::in|ios_base::out)
- int pubsync ()

Protected Member Functions

- void **__safe_gbump** (streamsize __n)
- void **__safe_pbump** (streamsize __n)
- void **_M_allocate_internal_buffer** ()
- bool **_M_convert_to_external** (char_type *, streamsize)
- void _M_create_pback ()
- void **_M_destroy_internal_buffer** () throw ()
- void _M_destroy_pback () throw ()
- int **_M_get_ext_pos** (__state_type &__state)
- pos_type **_M_seek** (off_type __off, ios_base::seekdir __way, __state_type __state)
- void _M_set_buffer (streamsize __off)
- bool **_M_terminate_output** ()
- void gbump (int __n)
- virtual void imbue (const locale &__loc)
- virtual int_type overflow (int_type __c=_Traits::eof())
- virtual int_type pbackfail (int_type __c=_Traits::eof())
- void pbump (int __n)
- virtual pos_type seekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)
- virtual pos_type seekpos (pos_type __pos, ios_base::openmode __mode=ios_base::in|ios_base::out)
- virtual __streambuf_type * setbuf (char_type *__s, streamsize __n)
- void setg (char_type *__gbeg, char_type *__gnext, char_type *__gend)
- void setp (char_type *__pbeg, char_type *__pend)
- virtual streamsize showmanyc ()
- void **swap** (basic_streambuf &__sb)
- virtual int sync ()
- virtual int_type uflow ()
- virtual int_type underflow ()
- virtual streamsize xsgetn (char_type *__s, streamsize __n)
- virtual streamsize xsputn (const char_type *__s, streamsize __n)

- char_type * eback () const

- char_type * gptr () const
- char_type * egptr () const

- char_type * pbase () const
- char_type * pptr () const
- char_type * epptr () const

Protected Attributes

- char_type * _M_buf
- bool **_M_buf_allocated**
- locale _M_buf_locale
- size_t _M_buf_size
- const __codecvt_type * **_M_codecvt**
- char * _M_ext_buf
- streamsize _M_ext_buf_size
- char * **_M_ext_end**
- const char * _M_ext_next
- __file_type **_M_file**
- char_type * _M_in_beg
- char_type * _M_in_cur
- char_type * _M_in_end
- __c_lock **_M_lock**
- ios_base::openmode _M_mode
- char_type * _M_out_beg
- char_type * _M_out_cur
- char_type * _M_out_end
- bool _M_reading
- __state_type **_M_state_beg**
- __state_type **_M_state_cur**
- __state_type **_M_state_last**
- bool **_M_writing**

- char_type _M_pback
- char_type * _M_pback_cur_save
- char_type * _M_pback_end_save
- bool _M_pback_init

Friends

- class **ios_base**

5.217.1 Detailed Description

template<typename _CharT, typename _Traits>
class std::basic_filebuf< _CharT, _Traits >

The actual work of input and output (for files).

Template Parameters

_CharT	Type of character stream.
_Traits	Traits for character type, defaults to char_traits<_CharT>.

This class associates both its input and output sequence with an external disk file, and maintains a joint file position for both sequences. Many of its semantics are described in terms of similar behavior in the Standard C Library's `FILE` streams.
Requirements on traits_type, specific to this class:

- traits_type::pos_type must be fpos<traits_type::state_type>
- traits_type::off_type must be streamoff
- traits_type::state_type must be Assignable and DefaultConstructible,
- traits_type::state_type() must be the initial state for codecvt.

5.217.2 Constructor & Destructor Documentation

basic_filebuf()

```
template<typename _CharT, typename _Traits>
std::basic_filebuf< _CharT, _Traits >::basic_filebuf ()
```

Does not open any files.
The default constructor initializes the parent class using its own default ctor.
References _M_buf, std::basic_streambuf< _CharT, _Traits >::_M_buf_locale, _M_buf_size, _M_ext_buf, _M_ext_buf_size, _M_ext_next, _M_mode, _M_pback, _M_pback_cur_save, _M_pback_end_save, _M_pback_init, and _M_reading.
Referenced by __gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf(), and close().

~basic_filebuf()

```
template<typename _CharT, typename _Traits>
virtual std::basic_filebuf< _CharT, _Traits >::~basic_filebuf ()  [inline], [virtual]
```

The destructor closes the file first.

5.217.3 Member Function Documentation

_M_create_pback()

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf< _CharT, _Traits >::_M_create_pback ()  [inline], [protected]
```

Initializes pback buffers, and moves normal buffers to safety. Assumptions: _M_in_cur has already been moved back
Referenced by pbackfail().

_M_destroy_pback()

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf< _CharT, _Traits >::_M_destroy_pback () throw ()  [inline], [protected]
```

Deactivates pback buffer contents, and restores normal buffer. Assumptions: The pback buffer has only moved forward.
Referenced by overflow(), seekoff(), seekpos(), underflow(), and xsgetn().

_M_set_buffer()

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf< _CharT, _Traits >::_M_set_buffer (
            streamsize __off)  [inline], [protected]
```

This function sets the pointers of the internal buffer, both get and put areas. Typically:
__off == egptr() - eback() upon underflow/uflow (**read** mode); __off == 0 upon overflow (**write** mode); __off == -1 upon open, setbuf, seekoff/pos (**uncommitted** mode).
NB: epptr() - pbase() == _M_buf_size - 1, since _M_buf_size reflects the actual allocated memory and the last cell is reserved for the overflow char of a full put area.
Referenced by __gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf(), __gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf(), close(), imbue(), open(), overflow(), pbackfail(), underflow(), xsgetn(), and xsputn().

close()

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::__filebuf_type * std::basic_filebuf< _CharT, _Traits >::close
()
```

Closes the currently associated file.

Returns

`this` on success, NULL on failure

If no file is currently open, this function immediately fails.
If a *put buffer area* exists, `overflow(eof)` is called to flush all the characters. The file is then closed.
If any operations fail, this function also fails.
References basic_filebuf(), _M_mode, _M_pback_init, _M_reading, _M_set_buffer(), and is_open().
Referenced by open().

eback()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::eback () const  [inline], [protected], [inherited]
```

Access to the get area.
These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_stringbuf< _Cha std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_filebuf< _CharT, _Traits >::underflow(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

egptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::egptr () const  [inline], [protected], [inherited]
```

Access to the get area.
These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Referenced by std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_filebuf< _CharT, _Traits >::underflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), std::wbuffer_convert< _Codecvt, _Elem, _Tr >::underflow(), std::basic_filebuf< _CharT, _Traits >::xsgetn(), and xsgetn().

epptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::epptr () const  [inline], [protected], [inherited]
```

Access to the put area.
These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_filebuf< _CharT, _Traits >::xsputn(), and xsputn().

gbump()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::gbump (
            int __n)  [inline], [protected], [inherited]
```

Moving the read position.

Parameters

_↩ _n	The delta by which to move.

This just advances the read position without returning any data.
Referenced by std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

getloc()

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::getloc () const  [inline], [inherited]
```

Locale access.

Returns

The current locale in effect.

If pubimbue(loc) has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

gptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::gptr () const  [inline], [protected], [inherited]
```

Access to the get area.
These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_stringbuf< _Cha std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_filebuf< _CharT, _Traits >::underflow(), std::basic_stringbuf< _CharT, _T std::wbuffer_convert< _Codecvt, _Elem, _Tr >::underflow(), std::basic_filebuf< _CharT, _Traits >::xsgetn(), and xsgetn().

imbue()

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf< _CharT, _Traits >::imbue (
            const locale & __loc)  [protected], [virtual]
```

Changes translations.

Parameters

__loc	A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

Note

Base class version does nothing.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_ext_buf, _M_ext_next, _M_mode, _M_reading, _M_set_buffer(), std::ios_base::cur, std::basic_streambuf< _CharT, _Trait std::basic_streambuf< _CharT, _Traits >::gptr(), is_open(), and seekoff().

in_avail()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::in_avail ()  [inline], [inherited]
```

Looking ahead into the stream.

Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

is_open()

```
template<typename _CharT, typename _Traits>
bool std::basic_filebuf< _CharT, _Traits >::is_open () const throw ( )   [inline], [nodiscard]
```

Returns true if the external file is open.
Referenced by __gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf(), __gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf(), close(), imbue(), open(), seekoff(), seekpos(), setbuf(), and showmanyc().

open() [1/3]

```
template<typename _CharT, typename _Traits>
template<typename _Path>
_If_fs_path< _Path, __filebuf_type * > std::basic_filebuf< _CharT, _Traits >::open (
            const _Path & __s,
            ios_base::openmode __mode)  [inline]
```

Opens an external file.

Parameters

__s	The name of the file, as a filesystem::path.
__mode	The open mode flags.

Returns

`this` on success, NULL on failure

open() [2/3]

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::__filebuf_type * std::basic_filebuf< _CharT, _Traits >::open (
            const char * __s,
            ios_base::openmode __mode)
```

Opens an external file.

Parameters

__s	The name of the file.
__mode	The open mode flags.

Returns

`this` on success, NULL on failure

If a file is already open, this function immediately fails. Otherwise it tries to open the file named *__s* using the flags given in *__mode*.
Table 92, adapted here, gives the relation between openmode combinations and the equivalent `fopen()` flags. (NB: lines app, in|out|app, in|app, binary|app, binary|in|out|app, and binary|in|app per DR 596)

ios_base Flag combination				stdio equivalent
binary	in	out	trunc	app	
		+			w
		+	+		a
			+		a
		+	+		w
	+				r
	+	+			r+
	+	+	+		w+
	+	+	+		a+
	+		+		a+
	+				wb
	+		+		ab
	+		+		ab
	+		+		wb
	+	+			rb
	+	+			r+b
	+	+	+		w+b
	+	+	+		a+b
	+	+			a+b

References _M_mode, _M_reading, _M_set_buffer(), std::ios_base::ate, close(), std::ios_base::end, is_open(), and seekoff().

open() [3/3]

```
template<typename _CharT, typename _Traits>
__filebuf_type * std::basic_filebuf< _CharT, _Traits >::open (
            const std::string & __s,
            ios_base::openmode __mode)  [inline]
```

Opens an external file.

Parameters

__s	The name of the file.
__mode	The open mode flags.

Returns

`this` on success, NULL on failure

overflow()

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::int_type std::basic_filebuf< _CharT, _Traits >::overflow (
            int_type __c = _Traits::eof())  [protected], [virtual]
```

Consumes data from the buffer; writes to the controlled sequence.

Parameters

_↩ _c	An additional character to consume.

Returns

eof() to indicate failure, something else (usually *__c*, or not_eof())

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if *__c* is not `eof()`.
For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.
A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_buf_size, _M_destroy_pback(), _M_mode, _M_reading, _M_set_buffer(), std::ios_base::app, std::ios_base::cur, std::ios_base::out, std::basic_streambuf< _CharT, _Traits >::pbase(), std::basic_streambuf< _CharT, _Traits >::pbum and std::basic_streambuf< _CharT, _Traits >::pptr().
Referenced by pbackfail(), sync(), underflow(), and xsgetn().

pbackfail()

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::int_type std::basic_filebuf< _CharT, _Traits >::pbackfail (
            int_type __c = _Traits::eof())  [protected], [virtual]
```

Tries to back up the input sequence.

Parameters

_↩ _c	The character to be inserted back into the sequence.

Returns

eof() on failure, *some other value* on success

Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_create_pback(), _M_mode, _M_pback_init, _M_reading, _M_set_buffer(), std::ios_base::cur, std::basic_streambuf< _CharT, _Traits >::eback(), std::basic_streambuf< _CharT, _Traits >::gbump(), std::basic_streambuf< _CharT, _
std::ios_base::in, overflow(), seekoff(), and underflow().

pbase()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::pbase () const  [inline], [protected], [inherited]
```

Access to the put area.
These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, std::basic_filebuf< _CharT, _Traits >::sync(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

pbump()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::pbump (
            int __n)  [inline], [protected], [inherited]
```

Moving the write position.

Parameters

_↩ _n	The delta by which to move.

This just advances the write position without returning any data.
Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow().

pptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::pptr () const  [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_filebuf< _CharT, _T std::basic_filebuf< _CharT, _Traits >::xsputn(), and xsputn().

pubimbue()

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::pubimbue (
            const locale & __loc)  [inline], [inherited]
```

Entry point for imbue().

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls the derived imbue(__loc).

pubseekoff()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekoff (
            off_type __off,
            ios_base::seekdir __way,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [inline], [inherited]
```

Alters the stream position.

Parameters

__off	Offset.
__way	Value for ios_base::seekdir.
__mode	Value for ios_base::openmode.

Calls virtual seekoff function.

pubseekpos()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekpos (
            pos_type __sp,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [inline], [inherited]
```

Alters the stream position.

Parameters

__sp	Position
__mode	Value for ios_base::openmode.

Calls virtual seekpos function.

pubsetbuf()

```
template<typename _CharT, typename _Traits>
basic_streambuf * std::basic_streambuf< _CharT, _Traits >::pubsetbuf (
            char_type * __s,
            streamsize __n)  [inline], [inherited]
```

Entry points for derived buffer functions.
The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

pubsync()

```
template<typename _CharT, typename _Traits>
int std::basic_streambuf< _CharT, _Traits >::pubsync ()  [inline], [inherited]
```

Calls virtual sync function.
Referenced by std::basic_istream< _CharT, _Traits >::sync().

sbumpc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sbumpc ()  [inline], [inherited]
```

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.
Referenced by std::basic_istream< _CharT, _Traits >::getline().

seekoff()

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::pos_type std::basic_filebuf< _CharT, _Traits >::seekoff (
            off_type ,
            ios_base::seekdir ,
            ios_base::openmode  = ios_base::in | ios_base::out)  [protected], [virtual]
```

Alters the stream positions.
Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_destroy_pback(), _M_reading, std::ios_base::cur, is_open(), std::basic_streambuf< _CharT, _Traits >::pbase(), and std::basic_streambuf< _CharT, _Traits >::pptr().
Referenced by imbue(), open(), and pbackfail().

seekpos()

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::pos_type std::basic_filebuf< _CharT, _Traits >::seekpos (
            pos_type ,
            ios_base::openmode  = ios_base::in | ios_base::out)  [protected], [virtual]
```

Alters the stream positions.
Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_destroy_pback(), std::ios_base::beg, and is_open().

setbuf()

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::__streambuf_type * std::basic_filebuf< _CharT, _Traits >↩
::setbuf (
            char_type * __s,
            streamsize __n)  [protected], [virtual]
```

Manipulates the buffer.

Parameters

_↩ _s	Pointer to a buffer area.
_↩ _n	Size of __s.

Returns

`this`

If no file has been opened, and both __s and __n are zero, then the stream becomes unbuffered. Otherwise, __s is used as a buffer; see https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.↩ html#io.streambuf.buffering for more.
Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_buf, _M_buf_size, and is_open().

setg()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setg (
            char_type * __gbeg,
            char_type * __gnext,
            char_type * __gend)  [inline], [protected], [inherited]
```

Setting the three read area pointers.

Parameters

__gbeg	A pointer.
__gnext	A pointer.
__gend	A pointer.

Postcondition

__gbeg == `eback()`, *__gnext* == `gptr()`, and *__gend* == `egptr()`

Referenced by std::wbuffer_convert< _Codecvt, _Elem, _Tr >::wbuffer_convert(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflo std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

setp()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setp (
            char_type * __pbeg,
            char_type * __pend)  [inline], [protected], [inherited]
```

Setting the three write area pointers.

Parameters

__pbeg	A pointer.
__pend	A pointer.

Postcondition

__pbeg == `pbase()`, *__pbeg* == `pptr()`, and *__pend* == `epptr()`

Referenced by std::wbuffer_convert< _Codecvt, _Elem, _Tr >::wbuffer_convert().

sgetc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sgetc ()  [inline], [inherited]
```

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.
Referenced by std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), and std::basic_istream< _CharT, _Traits >::ignore().

sgetn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sgetn (
            char_type * __s,
            streamsize __n)  [inline], [inherited]
```

Entry point for xsgetn.

Parameters

__s	A buffer area.
__n	A count.

Returns xsgetn(__s,__n). The effect is to fill *__s*[0] through *__s*[__n-1] with characters from the input sequence, if possible.

showmanyc()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_filebuf< _CharT, _Traits >::showmanyc ()  [protected], [virtual]
```

Investigating the data available.

Returns

An estimate of the number of characters available in the input sequence, or -1.

If it returns a positive value, then successive calls to `underflow()` *will not return* `traits::eof()` *until at least that number of characters have been supplied. If* `showmanyc()` *returns -1, then calls to* `underflow()` *or* `uflow()` *will fail.* [27.5.2.4.3]/1

Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to underflow or uflow] will not return* `eof()` *but that they will return immediately.*

The standard adds that *the morphemes of* `showmanyc` *are* ***es-how-many-see****, not* ***show-manic****.*

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_mode, std::ios_base::binary, std::basic_streambuf< _CharT, _Traits >::egptr(), std::basic_streambuf< _CharT, _Traits > std::ios_base::in, and is_open().

snextc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::snextc ()  [inline], [inherited]
```

Getting the next character.

Returns

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.
Referenced by std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< char_type, traits_type >::ignore(), std::basic_istream< char_type, traits_type >::operator>>(), and std::basic_istream< char_type, traits_type >::putback().

sputbackc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputbackc (
            char_type __c)  [inline], [inherited]
```

Pushing characters back into the input stream.

Parameters

__c	The character to push back.

Returns

The previous character, if possible.

Similar to sungetc(), but *__c* is pushed onto the stream instead of *the previous character.* If successful, the next character fetched from the input stream will be *__c*.
Referenced by std::basic_istream< _CharT, _Traits >::putback().

sputc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputc (
            char_type __c)  [inline], [inherited]
```

Entry point for all single-character output functions.

Parameters

_↩ _c	A character to output.

Returns

__c, if possible.

One of two public output functions.
If a write position is available for the output sequence (i.e., the buffer is not full), stores __c in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(↩ __c)`.
Referenced by std::wbuffer_convert< _Codecvt, _Elem, _Tr >::overflow().

sputn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sputn (
            const char_type * __s,
            streamsize __n)  [inline], [inherited]
```

Entry point for all single-character output functions.

Parameters

_↩ _s	A buffer read area.
_↩ _n	A count.

One of two public output functions.
Returns xsputn(__s,__n). The effect is to write __s[0] through __s[__n-1] to the output sequence, if possible.

sungetc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sungetc ()  [inline], [inherited]
```

Moving backwards in the input stream.

Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns pbackfail(). The effect is to *unget* the last character *gotten*.
Referenced by std::basic_istream< char_type, traits_type >::sentry::sentry().

sync()

```
template<typename _CharT, typename _Traits>
int std::basic_filebuf< _CharT, _Traits >::sync ()  [protected], [virtual]
```

Synchronizes the buffer arrays with the controlled sequences.

Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

Note

Base class version does nothing, returns zero.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References overflow(), std::basic_streambuf< _CharT, _Traits >::pbase(), and std::basic_streambuf< _CharT, _Traits >::pptr().

uflow()

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::uflow ()  [inline], [protected], [virtual],
[inherited]
```

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.
Reimplemented in __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >.
Referenced by xsgetn().

underflow()

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::int_type std::basic_filebuf< _CharT, _Traits >::underflow ()
[protected], [virtual]
```

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.
For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.
A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_buf_size, _M_destroy_pback(), _M_mode, _M_set_buffer(), std::basic_streambuf< _CharT, _Traits >::eback(), std::basic_streambuf< _CharT, _Traits >::egptr(), std::basic_streambuf< _CharT, _Traits >::gptr(), std::ios_base::in, and overflow().
Referenced by pbackfail().

xsgetn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_filebuf< _CharT, _Traits >::xsgetn (
            char_type * __s,
            streamsize __n)  [protected], [virtual]
```

Multiple character extraction.

Parameters

_↩ _s	A buffer area.
_↩ _n	Maximum number of characters to assign.

Returns

The number of characters assigned.

Fills __s[0] through __s[__n-1] with characters from the input sequence, as if by `sbumpc()`. Stops when either __n characters have been copied, or when `traits::eof()` would be copied.
It is expected that derived classes provide a more efficient implementation by overriding this definition.
Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_buf_size, _M_destroy_pback(), _M_mode, _M_pback_init, _M_reading, _M_set_buffer(), std::basic_streambuf< _CharT, _
std::basic_streambuf< _CharT, _Traits >::egptr(), std::basic_streambuf< _CharT, _Traits >::gbump(), std::basic_streambuf< _CharT, _T
std::ios_base::in, overflow(), std::basic_streambuf< _CharT, _Traits >::setg(), and std::basic_streambuf< char_type, traits_type >::xsgetn

xsputn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_filebuf< _CharT, _Traits >::xsputn (
            const char_type * __s,
            streamsize __n)  [protected], [virtual]
```

Multiple character insertion.

Parameters

_↩ _s	A buffer area.
_↩ _n	Maximum number of characters to write.

Returns

The number of characters written.

Writes __s[0] through __s[__n-1] to the output sequence, as if by `sputc()`. Stops when either *n* characters have been copied, or when `sputc()` would return `traits::eof()`.
It is expected that derived classes provide a more efficient implementation by overriding this definition.
Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_buf_size, _M_mode, _M_reading, _M_set_buffer(), std::ios_base::app, std::basic_streambuf< _CharT, _Traits >::epptr(),
std::ios_base::out, std::basic_streambuf< _CharT, _Traits >::pbase(), std::basic_streambuf< _CharT, _Traits >::pptr(),
and std::basic_streambuf< char_type, traits_type >::xsputn().

5.217.4 Member Data Documentation

_M_buf

```
template<typename _CharT, typename _Traits>
char_type* std::basic_filebuf< _CharT, _Traits >::_M_buf  [protected]
```

Pointer to the beginning of internal buffer.
Referenced by basic_filebuf(), and setbuf().

_M_buf_locale

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::_M_buf_locale  [protected], [inherited]
```

Current locale setting.
Referenced by std::basic_filebuf< _CharT, _Traits >::basic_filebuf().

_M_buf_size

```
template<typename _CharT, typename _Traits>
size_t std::basic_filebuf< _CharT, _Traits >::_M_buf_size  [protected]
```

Actual size of internal buffer. This number is equal to the size of the put area + 1 position, reserved for the overflow char of a full area.
Referenced by basic_filebuf(), __gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf(), __gnu_cxx::stdio_filebuf< _CharT, _Traits >::s overflow(), setbuf(), underflow(), xsgetn(), and xsputn().

_M_ext_buf

```
template<typename _CharT, typename _Traits>
char* std::basic_filebuf< _CharT, _Traits >::_M_ext_buf  [protected]
```

Buffer for external characters. Used for input when codecvt::always_noconv() == false. When valid, this corresponds to eback().
Referenced by basic_filebuf(), and imbue().

_M_ext_buf_size

```
template<typename _CharT, typename _Traits>
streamsize std::basic_filebuf< _CharT, _Traits >::_M_ext_buf_size  [protected]
```

Size of buffer held by _M_ext_buf.
Referenced by basic_filebuf().

_M_ext_next

```
template<typename _CharT, typename _Traits>
const char* std::basic_filebuf< _CharT, _Traits >::_M_ext_next  [protected]
```

Pointers into the buffer held by _M_ext_buf that delimit a subsequence of bytes that have been read but not yet converted. When valid, _M_ext_next corresponds to egptr().
Referenced by basic_filebuf(), and imbue().

_M_in_beg

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_beg  [protected], [inherited]
```

Start of get area.

_M_in_cur

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_cur  [protected], [inherited]
```
Current read area.

_M_in_end

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_end  [protected], [inherited]
```
End of get area.

_M_mode

```
template<typename _CharT, typename _Traits>
ios_base::openmode std::basic_filebuf< _CharT, _Traits >::_M_mode  [protected]
```
Place to stash in || out || in | out settings for current filebuf.
Referenced by basic_filebuf(), __gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf(), __gnu_cxx::stdio_filebuf< _CharT, _Traits >::
close(), imbue(), open(), overflow(), pbackfail(), showmanyc(), underflow(), xsgetn(), and xsputn().

_M_out_beg

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_beg  [protected], [inherited]
```
Start of put area.

_M_out_cur

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_cur  [protected], [inherited]
```
Current put area.

_M_out_end

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_end  [protected], [inherited]
```
End of put area.

_M_pback

```
template<typename _CharT, typename _Traits>
char_type std::basic_filebuf< _CharT, _Traits >::_M_pback  [protected]
```
Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Referenced by basic_filebuf().

_M_pback_cur_save

```
template<typename _CharT, typename _Traits>
char_type* std::basic_filebuf< _CharT, _Traits >::_M_pback_cur_save  [protected]
```
Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Referenced by basic_filebuf().

_M_pback_end_save

```
template<typename _CharT, typename _Traits>
char_type* std::basic_filebuf< _CharT, _Traits >::_M_pback_end_save  [protected]
```

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Referenced by basic_filebuf().

_M_pback_init

```
template<typename _CharT, typename _Traits>
bool std::basic_filebuf< _CharT, _Traits >::_M_pback_init  [protected]
```

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Referenced by basic_filebuf(), close(), pbackfail(), and xsgetn().

_M_reading

```
template<typename _CharT, typename _Traits>
bool std::basic_filebuf< _CharT, _Traits >::_M_reading  [protected]
```

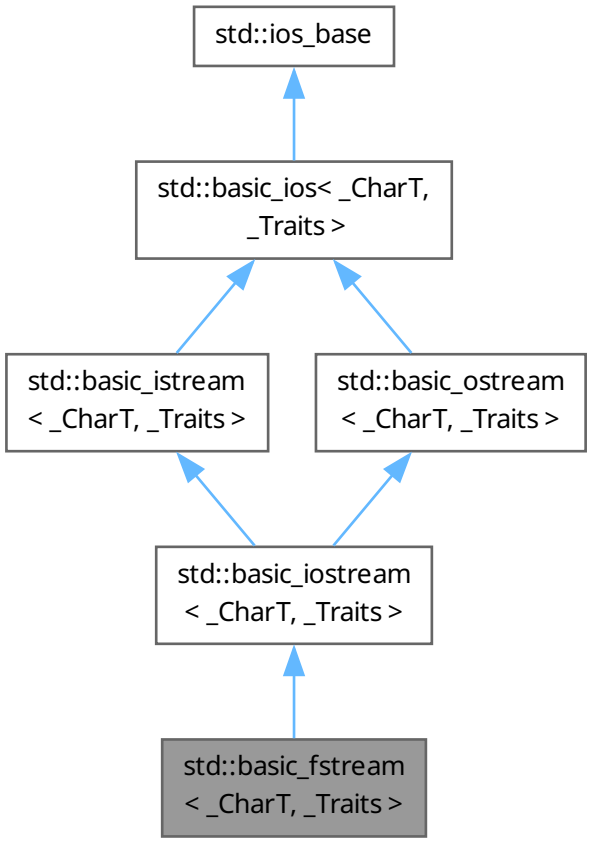
_M_reading == false && _M_writing == false for **uncommitted** mode; _M_reading == true for **read** mode; _M_writing == true for **write** mode;
NB: _M_reading == true && _M_writing == true is unused.
Referenced by basic_filebuf(), __gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf(), __gnu_cxx::stdio_filebuf< _CharT, _Traits >::s close(), imbue(), open(), overflow(), pbackfail(), seekoff(), xsgetn(), and xsputn().
The documentation for this class was generated from the following files:

- fstream
- fstream.tcc

5.218 std::basic_fstream< _CharT, _Traits > Class Template Reference

```
#include <fstream>
```

Inheritance diagram for std::basic_fstream< _CharT, _Traits >:



Public Types

- typedef ctype< _CharT > **__ctype_type**
- typedef ctype< _CharT > **__ctype_type**
- typedef basic_filebuf< char_type, traits_type > **__filebuf_type**
- typedef basic_ios< char_type, traits_type > **__ios_type**
- typedef basic_iostream< char_type, traits_type > **__iostream_type**
- typedef basic_istream< _CharT, _Traits > **__istream_type**
- typedef num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > **__num_get_type**
- typedef num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > **__num_put_type**
- typedef basic_ostream< _CharT, _Traits > **__ostream_type**
- typedef basic_streambuf< _CharT, _Traits > **__streambuf_type**
- typedef basic_streambuf< _CharT, _Traits > **__streambuf_type**
- typedef _CharT **char_type**
- enum event { **erase_event** , **imbue_event** , **copyfmt_event** }
- typedef void(* event_callback) (event __e, ios_base &__b, int __i)
- typedef traits_type::int_type **int_type**

- typedef _Ios_Iostate iostate
- typedef traits_type::off_type **off_type**
- typedef _Ios_Openmode openmode
- typedef traits_type::pos_type **pos_type**
- typedef _Ios_Seekdir seekdir
- typedef _Traits **traits_type**

Public Member Functions

- basic_fstream ()
- **basic_fstream** (basic_fstream &&__rhs)
- template<typename _Path, typename _Require = _If_fs_path<_Path>>
  basic_fstream (const _Path &__s, ios_base::openmode __mode=ios_base::in|ios_base::out)
- **basic_fstream** (const basic_fstream &)=delete
- basic_fstream (const char *__s, ios_base::openmode __mode=ios_base::in|ios_base::out)
- basic_fstream (const std::string &__s, ios_base::openmode __mode=ios_base::in|ios_base::out)
- ~basic_fstream ()
- template<typename _ValueT>
  basic_istream< _CharT, _Traits > & **_M_extract** (_ValueT &__v)
- const locale & _M_getloc () const
- template<typename _ValueT>
  basic_ostream< _CharT, _Traits > & **_M_insert** (_ValueT __v)
- void **_M_setstate** (iostate __state)
- bool bad () const
- void clear (iostate __state=goodbit)
- void close ()
- basic_ios & copyfmt (const basic_ios &__rhs)
- bool eof () const
- iostate exceptions () const
- void exceptions (iostate __except)
- bool fail () const
- char_type fill () const
- char_type fill (char_type __ch)
- fmtflags flags () const
- fmtflags flags (fmtflags __fmtfl)
- __ostream_type & flush ()
- streamsize gcount () const
- basic_istream< char > & getline (char_type *__s, streamsize __n, char_type __delim)
- basic_istream< wchar_t > & **getline** (char_type *__s, streamsize __n, char_type __delim)
- locale getloc () const
- bool good () const
- basic_istream< char > & **ignore** (streamsize __n)
- basic_istream< wchar_t > & **ignore** (streamsize __n)
- basic_istream< char > & **ignore** (streamsize __n, int_type __delim)
- basic_istream< wchar_t > & **ignore** (streamsize __n, int_type __delim)
- locale imbue (const locale &__loc)
- bool is_open ()
- bool **is_open** () const
- long & iword (int __ix)
- char narrow (char_type __c, char __dfault) const

- template<typename _Path>
  _If_fs_path< _Path, void > open (const _Path &__s, ios_base::openmode __mode=ios_base::in|ios_base::out)
- void open (const char *__s, ios_base::openmode __mode=ios_base::in|ios_base::out)
- void open (const std::string &__s, ios_base::openmode __mode=ios_base::in|ios_base::out)
- __ostream_type & operator<< (__streambuf_type *__sb)
- __ostream_type & operator<< (const void *__p)
- __ostream_type & **operator**<< (nullptr_t)
- basic_fstream & **operator=** (basic_fstream &&__rhs)
- basic_fstream & **operator=** (const basic_fstream &)=delete
- __istream_type & operator>> (__streambuf_type *__sb)
- __istream_type & operator>> (void *&__p)
- streamsize precision () const
- streamsize precision (streamsize __prec)
- void *& pword (int __ix)
- __filebuf_type * rdbuf () const
- basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > *__sb)
- iostate rdstate () const
- void register_callback (event_callback __fn, int __index)
- __ostream_type & seekp (off_type, ios_base::seekdir)
- __ostream_type & seekp (pos_type)
- fmtflags setf (fmtflags __fmtfl)
- fmtflags setf (fmtflags __fmtfl, fmtflags __mask)
- void setstate (iostate __state)
- void **swap** (basic_fstream &__rhs)
- pos_type tellp ()
- basic_ostream< _CharT, _Traits > * tie () const
- basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > *__tiestr)
- void unsetf (fmtflags __mask)
- char_type widen (char __c) const
- streamsize width () const
- streamsize width (streamsize __wide)

- __istream_type & operator>> (__istream_type &(*__pf)(__istream_type &))
- __istream_type & operator>> (__ios_type &(*__pf)(__ios_type &))
- __istream_type & operator>> (ios_base &(*__pf)(ios_base &))

Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type std::basic_istream::sentry with the second argument (noskipws) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, ios_base::badbit will be turned on in the stream's error state (without causing an ios_base::failure to be thrown) and the original exception will be rethrown if badbit is set in the exceptions mask.

- __istream_type & operator>> (bool &__n)
- __istream_type & operator>> (short &__n)
- __istream_type & operator>> (unsigned short &__n)
- __istream_type & operator>> (int &__n)
- __istream_type & operator>> (unsigned int &__n)
- __istream_type & operator>> (long &__n)
- __istream_type & operator>> (unsigned long &__n)

- __istream_type & operator>> (long long &__n)
- __istream_type & operator>> (unsigned long long &__n)

- __istream_type & operator>> (float &__f)
- __istream_type & operator>> (double &__f)
- __istream_type & operator>> (long double &__f)

Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type std::basic_istream::sentry with the second argument (noskipws) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by gcount().

If an exception is thrown during extraction, ios_base::badbit will be turned on in the stream's error state (without causing an ios_base::failure to be thrown) and the original exception will be rethrown if badbit is set in the exceptions mask.

- int_type get ()
- __istream_type & get (char_type &__c)
- __istream_type & get (char_type *__s, streamsize __n, char_type __delim)
- __istream_type & get (char_type *__s, streamsize __n)
- __istream_type & get (__streambuf_type &__sb, char_type __delim)
- __istream_type & get (__streambuf_type &__sb)
- __istream_type & getline (char_type *__s, streamsize __n, char_type __delim)
- __istream_type & getline (char_type *__s, streamsize __n)
- __istream_type & ignore (streamsize __n, int_type __delim)
- __istream_type & ignore (streamsize __n)
- __istream_type & ignore ()
- int_type peek ()
- __istream_type & read (char_type *__s, streamsize __n)
- streamsize readsome (char_type *__s, streamsize __n)
- __istream_type & putback (char_type __c)
- __istream_type & unget ()
- int sync ()
- pos_type tellg ()
- __istream_type & seekg (pos_type)
- __istream_type & seekg (off_type, ios_base::seekdir)

- operator bool () const
- bool operator! () const

- __ostream_type & operator<< (__ostream_type &(*__pf)(__ostream_type &))
- __ostream_type & operator<< (__ios_type &(*__pf)(__ios_type &))
- __ostream_type & operator<< (ios_base &(*__pf)(ios_base &))

Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type std::basic_ostream::sentry. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, ios_base::badbit will be turned on in the stream's error state without causing an ios_base::failure to be thrown. The original exception will then be rethrown.

- __ostream_type & operator<< (long __n)
- __ostream_type & operator<< (unsigned long __n)
- __ostream_type & operator<< (bool __n)
- __ostream_type & operator<< (short __n)
- __ostream_type & operator<< (unsigned short __n)
- __ostream_type & operator<< (int __n)
- __ostream_type & operator<< (unsigned int __n)
- __ostream_type & operator<< (long long __n)
- __ostream_type & operator<< (unsigned long long __n)

- __ostream_type & operator<< (double __f)
- __ostream_type & operator<< (float __f)
- __ostream_type & operator<< (long double __f)

Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type std::basic_ostream::sentry. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, ios_base::badbit will be turned on in the stream's error state. If badbit is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- __ostream_type & put (char_type __c)
- __ostream_type & write (const char_type *__s, streamsize __n)

Static Public Member Functions

- static bool sync_with_stdio (bool __sync=true)
- static int xalloc () throw ()

Public Attributes

- class __attribute((__abi_tag__("cxx11"))) failure typedef _Ios_Fmtflags fmtflags

Static Public Attributes

- static const openmode **__noreplace**
- static const fmtflags adjustfield
- static const openmode app
- static const openmode ate
- static const iostate badbit
- static const fmtflags basefield
- static const seekdir beg
- static const openmode binary
- static const fmtflags boolalpha
- static const seekdir cur
- static const fmtflags dec
- static const seekdir end
- static const iostate eofbit
- static const iostate failbit
- static const fmtflags fixed
- static const fmtflags floatfield
- static const iostate goodbit

- static const fmtflags hex
- static const openmode in
- static const fmtflags internal
- static const fmtflags left
- static const fmtflags oct
- static const openmode out
- static const fmtflags right
- static const fmtflags scientific
- static const fmtflags showbase
- static const fmtflags showpoint
- static const fmtflags showpos
- static const fmtflags skipws
- static const openmode trunc
- static const fmtflags unitbuf
- static const fmtflags uppercase

Protected Types

- enum { **_S_local_word_size** }

Protected Member Functions

- void **_M_cache_locale** (const locale &__loc)
- void **_M_call_callbacks** (event __ev) throw ()
- void **_M_dispose_callbacks** (void) throw ()
- template<typename _ValueT>
  __istream_type & **_M_extract** (_ValueT &__v)
- _Words & **_M_grow_words** (int __index, bool __iword)
- void **_M_init** () throw ()
- template<typename _ValueT>
  __ostream_type & **_M_insert** (_ValueT __v)
- void **_M_move** (ios_base &) noexcept
- void **_M_swap** (ios_base &__rhs) noexcept
- void init (basic_streambuf< _CharT, _Traits > *__sb)
- void **move** (basic_ios &&__rhs)
- void **move** (basic_ios &__rhs)
- void **set_rdbuf** (basic_streambuf< _CharT, _Traits > *__sb)
- void **swap** (basic_ios &__rhs) noexcept
- void **swap** (basic_iostream &__rhs)
- void **swap** (basic_istream &__rhs)
- void **swap** (basic_ostream &__rhs)

Protected Attributes

- _Callback_list * **_M_callbacks**
- const __ctype_type * **_M_ctype**
- iostate **_M_exception**
- char_type **_M_fill**
- bool **_M_fill_init**
- fmtflags **_M_flags**
- streamsize _M_gcount
- locale **_M_ios_locale**

- _Words **_M_local_word** [_S_local_word_size]
- const __num_get_type * **_M_num_get**
- const __num_put_type * **_M_num_put**
- streamsize **_M_precision**
- basic_streambuf< _CharT, _Traits > * **_M_streambuf**
- iostate **_M_streambuf_state**
- basic_ostream< _CharT, _Traits > * **_M_tie**
- streamsize **_M_width**
- _Words * **_M_word**
- int **_M_word_size**
- _Words **_M_word_zero**

5.218.1 Detailed Description

template<typename _CharT, typename _Traits>
class std::basic_fstream< _CharT, _Traits >

Controlling input and output for files.

Template Parameters

_CharT	Type of character stream.
_Traits	Traits for character type, defaults to char_traits<_CharT>.

This class supports reading from and writing to named files, using the inherited functions from std::basic_iostream. To control the associated sequence, an instance of std::basic_filebuf is used, which this page refers to as `sb`.

5.218.2 Member Typedef Documentation

event_callback

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i)  [inherited]
```

The type of an event callback function.

Parameters

_e	One of the members of the event enum.
_b	Reference to the ios_base object.
_i	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several ios_base and basic_ios functions, specifically imbue(), copyfmt(), and ~ios().

iostate

```
typedef _Ios_Iostate std::ios_base::iostate  [inherited]
```

This is a bitmask type.
_Ios_Iostate is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit

- eofbit
- failbit
- goodbit

openmode

`typedef _Ios_Openmode std::ios_base::openmode [inherited]`
This is a bitmask type.
_Ios_Openmode is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type openmode are:

- app
- ate
- binary
- in
- out
- trunc

seekdir

`typedef _Ios_Seekdir std::ios_base::seekdir [inherited]`
This is an enumerated type.
_Ios_Seekdir is implementation-defined. Defined values of type seekdir are:

- beg
- cur, equivalent to SEEK_CUR in the C standard library.
- end, equivalent to SEEK_END in the C standard library.

5.218.3 Member Enumeration Documentation

event

`enum std::ios_base::event [inherited]`
The set of events that may be passed to an event callback.
erase_event is used during ~ios() and copyfmt(). imbue_event is used during imbue(). copyfmt_event is used during copyfmt().

5.218.4 Constructor & Destructor Documentation

basic_fstream() [1/4]

```
template<typename _CharT, typename _Traits>
std::basic_fstream< _CharT, _Traits >::basic_fstream () [inline]
```
Default constructor.
Initializes sb using its default constructor, and passes &sb to the base class initializer. Does not open any files (you haven't given it a filename to open).

basic_fstream() [2/4]

```
template<typename _CharT, typename _Traits>
std::basic_fstream< _CharT, _Traits >::basic_fstream (
            const char * __s,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [inline], [explicit]
```

Create an input/output file stream.

Parameters

__s	Null terminated string specifying the filename.
__mode	Open file in specified mode (see std::ios_base).

basic_fstream() [3/4]

```
template<typename _CharT, typename _Traits>
std::basic_fstream< _CharT, _Traits >::basic_fstream (
            const std::string & __s,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [inline], [explicit]
```

Create an input/output file stream.

Parameters

__s	Null terminated string specifying the filename.
__mode	Open file in specified mode (see std::ios_base).

basic_fstream() [4/4]

```
template<typename _CharT, typename _Traits>
template<typename _Path, typename _Require = _If_fs_path<_Path>>
std::basic_fstream< _CharT, _Traits >::basic_fstream (
            const _Path & __s,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [inline]
```

Create an input/output file stream.

Parameters

__s	filesystem::path specifying the filename.
__mode	Open file in specified mode (see std::ios_base).

~basic_fstream()

```
template<typename _CharT, typename _Traits>
std::basic_fstream< _CharT, _Traits >::~basic_fstream ()  [inline]
```

The destructor does nothing.
The file is closed by the filebuf object, not the formatting stream.

5.218.5 Member Function Documentation

_M_getloc()

```
const locale & std::ios_base::_M_getloc () const  [inline], [inherited]
```

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.
Referenced by std::money_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter
std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIte
std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::num_put< _CharT, _OutIter
std::time_put< _CharT, _OutIter >::do_put(), std::time_get< _CharT, _InIter >::get(), and std::time_put< _CharT, _OutIter >::put().

bad()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry().

clear()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::clear (
            iostate __state = goodbit)  [inherited]
```

[Re]sets the error state.

Parameters

__state	The new state flag(s) to set.

See std::ios_base::iostate for the possible bit values. Most users will not need to pass an argument.
References std::ios_base::badbit, exceptions(), rdbuf(), and rdstate().
Referenced by std::basic_ios< char_type, traits_type >::exceptions(), std::__detail::operator>>(), std::basic_istream< _CharT, _Traits >
rdbuf(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char_type, traits
and std::basic_istream< _CharT, _Traits >::unget().

close()

```
template<typename _CharT, typename _Traits>
void std::basic_fstream< _CharT, _Traits >::close ()  [inline]
```

Close the file.
Calls `std::basic_filebuf::close()`. If that function fails, `failbit` is set in the stream's error state.

copyfmt()

```
template<typename _CharT, typename _Traits>
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
            const basic_ios< _CharT, _Traits > & __rhs)  [inherited]
```

Copies fields of __rhs into this.

Parameters

__rhs	The source values for the copies.

Returns

Reference to this object.

All fields of __rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase_event. After copying, each (new) callback is invoked with copyfmt_event. The final step is to copy exceptions().
References basic_ios(), std::__addressof(), exceptions(), fill(), std::ios_base::flags(), std::ios_base::getloc(), std::ios_base::precision(), tie(), std::tie(), and std::ios_base::width().

eof()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

exceptions() [1/2]

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions () const  [inline], [nodiscard], [inherited]
```

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.
Referenced by clear(), and copyfmt().

exceptions() [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
            iostate __except)  [inline], [inherited]
```

Throwing exceptions on errors.

Parameters

__except	The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type std::ios_base::failure is thrown.
If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
```

```
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr « "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr « "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.
Referenced by std::basic_ios< char_type, traits_type >::operator bool(), std::basic_ios< char_type, traits_type >::operator!(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >
std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::
and std::regex_traits< _Ch_type >::value().

fill() [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const  [inline], [nodiscard], [inherited]
```

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.
Referenced by copyfmt().

fill() [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
            char_type __ch)  [inline], [inherited]
```

Sets a new *empty* character.

Parameters

__ch	The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

flags() [1/2]

```
fmtflags std::ios_base::flags () const  [inline], [nodiscard], [inherited]
```
Access to format flags.

Returns

The format control flags for both input and output.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >
std::num_put< _CharT, _OutIter >::do_put(), std::num_put< _CharT, _OutIter >::do_put(), std::chrono::operator<<(),
std::operator<<(), std::__detail::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(),
std::operator>>(), std::operator>>(), and std::operator>>().

flags() [2/2]

```
fmtflags std::ios_base::flags (
            fmtflags __fmtfl)  [inline], [inherited]
```
Setting new format flags all at once.

Parameters

__fmtfl	The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *__fmtfl*.
References fmtflags.

flush()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::flush ()  [inherited]
```
Synchronizing the stream buffer.

Returns

*this

If `rdbuf()` is a null pointer, changes nothing.
Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit.
References std::ios_base::badbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits

gcount()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::gcount () const  [inline], [inherited]
```
Character counting.

Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

get() [1/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::get (
            void )  [inherited]
```

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().
References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

get() [2/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::get (
            __streambuf_type & __sb)  [inline], [inherited]
```

Extraction into another streambuf.

Parameters

__sb	A streambuf in which to store data.

Returns

*this

Returns get(__sb,widen('\n')).

get() [3/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
            __streambuf_type & __sb,
            char_type __delim)  [inherited]
```

Extraction into another streambuf.

Parameters

__sb	A streambuf in which to store data.
__delim	A "stop" character.

Returns

*this

Characters are extracted and inserted into *__sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals *__delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.
References _M_gcount, and std::ios_base::goodbit.

get() [4/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
            char_type & __c)  [inherited]
```

Simple extraction.

Parameters

_↩ _c	The character in which to store data.

Returns

*this

Tries to extract a character and store it in __c. If none are available, sets failbit and returns traits::eof().

Note

This function is not overloaded on signed char and unsigned char.

References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

get() [5/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::get (
            char_type * __s,
            streamsize __n)  [inline], [inherited]
```

Simple multiple-character extraction.

Parameters

_↩ _s	Pointer to an array.
_↩ _n	Maximum number of characters to store in *s*.

Returns

*this

Returns `get`(__s,__n,widen('\n')).

get() [6/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
            char_type * __s,
            streamsize __n,
            char_type __delim)  [inherited]
```

Simple multiple-character extraction.

Parameters

__s	Pointer to an array.
__n	Maximum number of characters to store in *__s*.
__delim	A "stop" character.

Returns

*this

Characters are extracted and stored into *__s* until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals *__delim*, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.
In any case, a null character is stored into the next location in the array.

Note

This function is not overloaded on signed char and unsigned char.

References _M_gcount, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_streambuf< _CharT, _Traits >:

getline() [1/3]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::getline (
            char_type * __s,
            streamsize __n)  [inline], [inherited]
```

String extraction.

Parameters

_↩ _s	A character array in which to store the data.
_↩ _n	Maximum number of characters to extract.

Returns

*this

Returns `getline`(__s,__n,widen('\n')).

getline() [2/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::getline (
            char_type * __s,
            streamsize __n,
            char_type __delim)  [inherited]
```

String extraction.

Parameters

__s	A character array in which to store the data.
__n	Maximum number of characters to extract.
__delim	A "stop" character.

Returns

*this

Extracts and stores characters into *__s* until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the *__s* array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)
In any case, a null character is stored in the next location in the array.
References _M_gcount, std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_streambuf< _CharT, _Traits >::sbumpc(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _Cha

getline() [3/3]

```
basic_istream< char > & std::basic_istream< char >::getline (
            char_type * __s,
            streamsize __n,
            char_type __delim)  [inherited]
```

Explicit specialization declarations, defined in src/istream.cc.

getloc()

```
locale std::ios_base::getloc () const  [inline], [nodiscard], [inherited]
```

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::money_put< _CharT, _OutIter >::do_put(), std::basic_ios< _CharT, _Tr
std::chrono::operator<<(), std::operator>>(), std::operator>>(), and std::ws().

good()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if no error flags are set.

A wrapper around rdstate.
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::__detail::operator>>().

ignore() [1/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
            void )  [inherited]
```

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().
References _M_gcount, std::ios_base::goodbit, and std::basic_ios< _CharT, _Traits >::rdbuf().

ignore() [2/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
            streamsize __n)  [inherited]
```

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().
References _M_gcount, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_streambuf< _CharT, _Traits >:

ignore() [3/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
            streamsize __n,
            int_type __delim)  [inherited]
```

Discarding characters.

Parameters

__n	Number of characters to discard.
__delim	A "stop" character.

Returns

*this

Extracts characters and throws them away until one of the following happens:

- if *__n* != `std::numeric_limits<int>::max()`, *__n* characters are extracted
- the input sequence reaches end-of-file
- the next character equals *__delim* (in this case, the character is extracted); note that this condition will never occur if *__delim* equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.
References _M_gcount, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_streambuf< _CharT, _Traits >:

imbue()

```
template<typename _CharT, typename _Traits>
locale std::basic_ios< _CharT, _Traits >::imbue (
            const locale & __loc)  [inherited]
```

Moves to a new locale.

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.
Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html
References std::ios_base::getloc(), std::ios_base::imbue(), and rdbuf().
Referenced by std::chrono::operator<<().

init()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::init (
            basic_streambuf< _CharT, _Traits > * __sb)  [protected], [inherited]
```

All setup is performed here.
This is called from the public constructor. It is not virtual and cannot be redefined.
Referenced by std::basic_ios< char_type, traits_type >::basic_ios(), and std::basic_ostream< char_type, traits_type >::basic_ostream().

is_open()

```
template<typename _CharT, typename _Traits>
bool std::basic_fstream< _CharT, _Traits >::is_open ()  [inline], [nodiscard]
```

Wrapper to test for an open file.

Returns

`rdbuf()->is_open()`

iword()

```
long & std::ios_base::iword (
            int __ix)  [inline], [inherited]
```

Access to integer array.

Parameters

__ix	Index into the array.

Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.
References iword().
Referenced by iword().

narrow()

```
template<typename _CharT, typename _Traits>
char std::basic_ios< _CharT, _Traits >::narrow (
            char_type __c,
            char __dfault) const  [inline], [inherited]
```

Squeezes characters.

Parameters

__c	The character to narrow.
__dfault	The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.
Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c,dfault)
```

Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html

open() [1/3]

```
template<typename _CharT, typename _Traits>
template<typename _Path>
_If_fs_path< _Path, void > std::basic_fstream< _CharT, _Traits >::open (
            const _Path & __s,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [inline]
```

Opens an external file.

Parameters

__s	The name of the file, as a filesystem::path.
__mode	The open mode flags.

Calls `std::basic_filebuf::open(__s,__mode)`. If that function fails, `failbit` is set in the stream's error state.

open() [2/3]

```
template<typename _CharT, typename _Traits>
void std::basic_fstream< _CharT, _Traits >::open (
```

```
            const char * __s,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [inline]
```

Opens an external file.

Parameters

__s	The name of the file.
__mode	The open mode flags.

Calls `std::basic_filebuf::open(__s,__mode)`. If that function fails, `failbit` is set in the stream's error state.

open() [3/3]

```
template<typename _CharT, typename _Traits>
void std::basic_fstream< _CharT, _Traits >::open (
            const std::string & __s,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [inline]
```

Opens an external file.

Parameters

__s	The name of the file.
__mode	The open mode flags.

Calls `std::basic_filebuf::open(__s,__mode)`. If that function fails, `failbit` is set in the stream's error state.

operator bool()

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool () const  [inline], [explicit], [nodiscard],
[inherited]
```

The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

operator"!()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator!  () const  [inline], [nodiscard], [inherited]
```

The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

operator<<() [1/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            __ios_type &(* __pf )(__ios_type &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

operator<<() [2/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            __ostream_type &(* __pf )(__ostream_type &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

operator<<() [3/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
            __streambuf_type * __sb)  [inherited]
```

Extracting from another streambuf.

Parameters

__sb	A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.
If `__sb` is NULL, the stream will set failbit in its error state.
Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets failbit in the error state

If the function inserts no characters, failbit is set.

operator<<() [4/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            bool __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [5/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            const void * __p)  [inline], [inherited]
```

Pointer arithmetic inserters.

Parameters

_↩ _*p*	A variable of pointer type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [6/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            double __f)  [inline], [inherited]
```

Floating point arithmetic inserters.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [7/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            float __f)  [inline], [inherited]
```

Floating point arithmetic inserters.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [8/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
            int __n)  [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _*n*	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [9/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            ios_base &(* __pf )(ios_base &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

operator<<() [10/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            long __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _*n*	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [11/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            long double __f)  [inline], [inherited]
```

Floating point arithmetic inserters.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [12/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            long long __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [13/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
            short __n)  [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.
References basic_ostream(), std::ios_base::badbit, std::ostreambuf_iterator< _CharT, _Traits >::failed(), std::ios_base::goodbit, std::num_put< _CharT, _OutIter >::put(), std::basic_ios< _CharT, _Traits >::setstate(), and std::use_facet().

operator<<() [14/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned int __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [15/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned long __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [16/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned long long __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [17/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned short __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator>>() [1/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            __ios_type &(* __pf )(__ios_type &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the iomanip header.

operator>>() [2/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            __istream_type &(* __pf )(__istream_type &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the iomanip header.

operator>>() [3/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
            __streambuf_type * __sb)  [inherited]
```

Extracting into another streambuf.

Parameters

__sb	A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.
If `__sb` is NULL, the stream will set failbit in its error state.
Characters are extracted from this stream and inserted into the `__sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.
References std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

operator>>() [4/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            bool & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [5/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            double & __f)  [inline], [inherited]
```

Floating point arithmetic extractors.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [6/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            float & __f)  [inline], [inherited]
```

Floating point arithmetic extractors.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [7/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
            int & __n)  [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.
References std::ios_base::badbit, std::ios_base::failbit, std::num_get< _CharT, _InIter >::get(), std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::setstate(), and std::use_facet().

operator>>() [8/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            ios_base &(* __pf )(ios_base &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as std::ws and std::dec use these functions in constructs like std::cin >> std::ws. For more information, see the iomanip header.

operator>>() [9/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            long & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [10/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            long double & __f)  [inline], [inherited]
```

Floating point arithmetic extractors.

Parameters

↩ _↩ ↩ _↩ f	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [11/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            long long & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [12/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
            short & __n)  [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.
References std::ios_base::badbit, std::ios_base::failbit, std::num_get< _CharT, _InIter >::get(), std::ios_base::goodbit, and std::use_facet().

operator>>() [13/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned int & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [14/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned long & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [15/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned long long & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [16/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned short & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [17/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            void *& __p)  [inline], [inherited]
```

Basic arithmetic extractors.

Parameters

_↩ _p	A variable of pointer type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

peek()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::peek (
            void )  [inherited]
```

Looking ahead in the stream.

Returns

The next character, or eof().

If, after constructing the sentry object, good() is false, returns traits::eof(). Otherwise reads but does not extract the next input character.

precision() [1/2]

```
streamsize std::ios_base::precision () const  [inline], [nodiscard], [inherited]
```

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::chrono::operator<<().

precision() [2/2]

```
streamsize std::ios_base::precision (
            streamsize __prec)  [inline], [inherited]
```

Changing flags.

Parameters

__prec	The new precision value.

Returns

The previous value of precision().

put()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (
            char_type __c)  [inherited]
```

Simple insertion.

Parameters

_↩ _c	The character to insert.

Returns

*this

Tries to insert __c.

Note

This function is not overloaded on signed char and unsigned char.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits

putback()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::putback (
            char_type __c)  [inherited]
```

Unextracting a single character.

Parameters

_↩ _c	The character to push back into the input stream.

Returns

*this

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.
If `rdbuf()` is null or if `sputbackc()` fails, sets badbit in the error state.

Note

This function first clears eofbit. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References _M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::rdstate(), std::basic_ios< _CharT, _Traits >::setstate(), and std::basic_streambuf< _CharT, _Traits >::sputbackc().

pword()

```
void *& std::ios_base::pword (
            int __ix)  [inline], [inherited]
```

Access to void pointer array.

Parameters

_↩ _*ix*	Index into the array.

Returns

A reference to a void* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

rdbuf() [1/2]

```
template<typename _CharT, typename _Traits>
__filebuf_type * std::basic_fstream< _CharT, _Traits >::rdbuf () const  [inline], [nodiscard]
```

Accessing the underlying buffer.

Returns

The current basic_filebuf buffer.

This hides both signatures of std::basic_ios::rdbuf().

rdbuf() [2/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
            basic_streambuf< _CharT, _Traits > * __sb)  [inherited]
```

Changing the underlying buffer.

Parameters

__sb	The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.
Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream     foo;         // or some other derived type
std::streambuf*  p = .....;

foo.ios::rdbuf(p);            // ios == basic_ios<char>
```

References clear().

rdstate()

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::rdstate () const  [inline], [nodiscard], [inherited]
```

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See std::ios_base::iostate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().
Referenced by std::basic_ios< char_type, traits_type >::bad(), clear(), std::basic_ios< char_type, traits_type >::eof(), std::basic_ios< char_type, traits_type >::fail(), std::basic_ios< char_type, traits_type >::good(), std::basic_istream< _CharT, _Traits >::
std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char_type, traits_type >
and std::basic_istream< _CharT, _Traits >::unget().

read()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::read (
            char_type * __s,
            streamsize __n)  [inherited]
```

Extraction without delimiters.

Parameters

_↩ _s	A character array.
_↩ _n	Maximum number of characters to store.

Returns

*this

If the stream state is `good()`, extracts characters and stores them into *__s* until one of the following happens:

- *__n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

Note

This function is not overloaded on signed char and unsigned char.

References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

readsome()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::readsome (
            char_type * __s,
            streamsize __n)  [inherited]
```

Extraction until the buffer is exhausted, but no more.

Parameters

_↩ _s	A character array.
_↩ _n	Maximum number of characters to store.

Returns

The number of characters extracted.

Extracts characters and stores them into *__s* depending on the number of characters remaining in the streambuf's buffer, `rdbuf()->in_avail()`, called `A` here:

- if `A == -1`, sets eofbit and extracts no characters
- if `A == 0`, extracts no characters
- if `A > 0`, extracts `min(A,n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.
References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::min(), std::basic_ios< _CharT, _Traits >::
and std::basic_ios< _CharT, _Traits >::setstate().

register_callback()

```
void std::ios_base::register_callback (
             event_callback __fn,
             int __index)  [inherited]
```

Add the callback __fn with parameter __index.

Parameters

__fn	The function to add.
__index	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.
References fmtflags.

seekg() [1/2]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
             off_type __off,
             ios_base::seekdir __dir)  [inherited]
```

Changing the current read position.

Parameters

__off	A file offset object.
__dir	The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(__off,__dir)`. If that function fails, sets failbit.

Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::basic_ios< _CharT, _Traits >::fail(
std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::in, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits
and std::basic_ios< _CharT, _Traits >::setstate().

seekg() [2/2]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
            pos_type __pos)  [inherited]
```

Changing the current read position.

Parameters

__pos	A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets failbit.

Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::basic_ios< _CharT, _Traits >::fail(
std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::in, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits
and std::basic_ios< _CharT, _Traits >::setstate().

seekp() [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
            off_type __off,
            ios_base::seekdir __dir)  [inherited]
```

Changing the current write position.

Parameters

__off	A file offset object.
__dir	The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off,dir)`. If that function fails, sets failbit.
References std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::out, std::basic_ios< _CharT, _Traits >::rdbuf(),
and std::basic_ios< _CharT, _Traits >::setstate().

seekp() [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
            pos_type __pos)  [inherited]
```

Changing the current write position.

Parameters

__pos	A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.
References std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::out, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

setf() [1/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl)  [inline], [inherited]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.
References fmtflags.
Referenced by std::__detail::operator>>().

setf() [2/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl,
            fmtflags __mask)  [inline], [inherited]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.
__mask	The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.
References fmtflags.

setstate()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
            iostate __state)  [inline], [inherited]
```

Sets additional flags in the error state.

Parameters

__state	The additional state flag(s) to set.

See std::ios_base::iostate for the possible bit values.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::getline(), std::getline(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _Cha std::operator>>(), std::operator>>(), std::operator>>(), std::tr2::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >: std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits > std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), and std::ws().

sync()

```
template<typename _CharT, typename _Traits>
int std::basic_istream< _CharT, _Traits >::sync (
            void )  [inherited]
```

Synchronizing the stream buffer.

Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.
Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit and returns -1.
Otherwise, returns 0.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_streambuf< _CharT, _Traits >::pubsync(), std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

sync_with_stdio()

```
static bool std::ios_base::sync_with_stdio (
            bool __sync = true)  [static], [inherited]
```

Interaction with the standard C I/O objects.

Parameters

__sync	Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary

tellg()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::pos_type std::basic_istream< _CharT, _Traits >::tellg (
            void )  [inherited]
```

Getting the current read position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type`(-1) to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0,cur,in)`.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with putback, unget and seekg, eofbit is not cleared first.

References std::ios_base::badbit, std::ios_base::cur, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::in, and std::basic_ios< _CharT, _Traits >::rdbuf().

tellp()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits >::pos_type std::basic_ostream< _CharT, _Traits >::tellp ()  [inherited]
```

Getting the current write position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type`(-1) to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0,cur,out)`.
References std::ios_base::cur, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::out, and std::basic_ios< _CharT, _Traits >::rdbuf()

tie() [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie () const  [inline],
[nodiscard], [inherited]
```

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and copyfmt().

tie() [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie (
            basic_ostream< _CharT, _Traits > * __tiestr)  [inline], [inherited]
```

Ties this stream to an output stream.

Parameters

__tiestr	The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

unget()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::unget (
            void )  [inherited]
```

Unextracting the previous character.

Returns

*this

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.
If `rdbuf()` is null or if `sungetc()` fails, sets badbit in the error state.

Note

This function first clears eofbit. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References _M_gcount, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::ios_base::goodbit, and std::basic_ios< _CharT, _Traits >::rdstate().
Referenced by std::__detail::operator>>().

unsetf()

```
void std::ios_base::unsetf (
            fmtflags __mask)  [inline], [inherited]
```

Clearing format flags.

Parameters

__mask	The flags to unset.

This function clears *__mask* in the format flags.
References fmtflags.

widen()

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
            char __c) const  [inline], [inherited]
```

Widens characters.

Parameters

_↩ _c	The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.
Returns the result of
`std::use_facet<ctype<char_type> >(getloc()).widen(c)`
Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html
Referenced by std::basic_ios< char_type, traits_type >::fill(), std::getline(), std::getline(), and std::tr2::operator>>().

width() [1/2]

```
streamsize std::ios_base::width () const  [inline], [nodiscard], [inherited]
```

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_put< _CharT, _OutIter >::do_put(), std::operator>>(), and std::operator>>().

width() [2/2]

```
streamsize std::ios_base::width (
             streamsize __wide)  [inline], [inherited]
```

Changing flags.

Parameters

__wide	The new width value.

Returns

The previous value of width().

write()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::write (
             const char_type * __s,
             streamsize __n)  [inherited]
```

Character string insertion.

Parameters

_↩ _s	The array to insert.
_↩ _n	Maximum number of characters to insert.

Returns

*this

Characters are copied from __s and inserted into the stream until one of the following happens:

- __n characters are inserted
- inserting into the output sequence fails (in this case, badbit will be set in the stream's error state)

Note

This function is not overloaded on signed char and unsigned char.

xalloc()

```
static int std::ios_base::xalloc () throw ( )   [static], [inherited]
```

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the iword and pword functions. The expectation is that an application calls xalloc in order to obtain an index in the iword and pword arrays that can be used without fear of conflict.
The implementation maintains a static variable that is incremented and returned on each invocation. xalloc is guaranteed to return an index that is safe to use in the iword and pword arrays.

5.218.6 Member Data Documentation

_M_gcount

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::_M_gcount  [protected], [inherited]
```

The number of characters extracted in the previous unformatted function; see gcount().
Referenced by get(), get(), get(), get(), getline(), ignore(), ignore(), ignore(), putback(), read(), readsome(), and unget().

adjustfield

```
const fmtflags std::ios_base::adjustfield  [static], [inherited]
```

A mask of left|right|internal. Useful for the 2-arg form of setf.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::internal(), std::left(), and std::right().

app

```
const openmode std::ios_base::app  [static], [inherited]
```

Seek to end before each write.
Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

ate

```
const openmode std::ios_base::ate  [static], [inherited]
```

Open and seek to end immediately after opening.
Referenced by std::basic_filebuf< _CharT, _Traits >::open().

badbit

const iostate std::ios_base::badbit [static], [inherited]
Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< char_type, traits_type >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::basic_ios< _CharT, _Traits >::clear(), std::basic_ostream< _CharT, _Trai std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_ostream< _CharT, _Traits >::opera std::basic_ostream< char_type, traits_type >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >: std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< char_type, traits_ty std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_istream< char_type, traits_type and std::ws().

basefield

const fmtflags std::ios_base::basefield [static], [inherited]
A mask of dec|oct|hex. Useful for the 2-arg form of setf.
Referenced by std::dec(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::hex(), and std::oct().

beg

const seekdir std::ios_base::beg [static], [inherited]
Request a seek relative to the beginning of the stream.
Referenced by std::basic_filebuf< _CharT, _Traits >::seekpos().

binary

const openmode std::ios_base::binary [static], [inherited]
Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.↩ filestreams.binary.
Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

boolalpha

const fmtflags std::ios_base::boolalpha [static], [inherited]
Insert/extract bool in alphabetic rather than numeric format.
Referenced by std::boolalpha(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::noboolalpha().

cur

const seekdir std::ios_base::cur [static], [inherited]
Request a seek relative to the current position within the sequence.
Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

dec

const fmtflags std::ios_base::dec [static], [inherited]
Converts integer input or generates integer output in decimal base.
Referenced by std::dec().

end

const seekdir std::ios_base::end [static], [inherited]
Request a seek relative to the current end of the sequence.
Referenced by std::basic_filebuf< _CharT, _Traits >::open(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff().

eofbit

const iostate std::ios_base::eofbit [static], [inherited]
Indicates that an input operation reached the end of an input sequence.
Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >:
std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(),
std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter
std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::ge
std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< char_type, traits_type >::g
std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::putback(),
std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< char_type, traits_type >::read(), std::basic_istream< _CharT, _Traits
std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< char_type, traits_ty
std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

failbit

const iostate std::ios_base::failbit [static], [inherited]
Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::num_get< _CharT, _InIter >::do_get(),
std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(),
std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::ge
std::basic_istream< char_type, traits_type >::get(), std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::oper
std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(),
std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::s
std::basic_ostream< _CharT, _Traits >::seekp(), and std::basic_ostream< _CharT, _Traits >::seekp().

fixed

const fmtflags std::ios_base::fixed [static], [inherited]
Generate floating-point output in fixed-point notation.
Referenced by std::fixed(), and std::hexfloat().

floatfield

const fmtflags std::ios_base::floatfield [static], [inherited]
A mask of scientific|fixed. Useful for the 2-arg form of `setf`.
Referenced by std::defaultfloat(), std::fixed(), std::hexfloat(), and std::scientific().

goodbit

const iostate std::ios_base::goodbit [static], [inherited]
Indicates all is well.
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::num_get< _CharT, _InIter >::do_get(),
std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::d
std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >:
std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(),
std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore()
std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ostream< _CharT, _Traits >

std::basic_ostream< char_type, traits_type >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::
std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits
std::basic_istream< char_type, traits_type >::seekg(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Trait
and std::ws().

hex

`const fmtflags std::ios_base::hex [static], [inherited]`
Converts integer input or generates integer output in hexadecimal base.
Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::hex().

in

`const openmode std::ios_base::in [static], [inherited]`
Open for input. Default for `ifstream` and fstream.
Referenced by std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_stringbuf< _CharT, _Traits, _
std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_filebuf< _CharT, _Traits >::underflow(), std::basic_stringbuf< _CharT, _Traits, _
and std::basic_filebuf< _CharT, _Traits >::xsgetn().

internal

`const fmtflags std::ios_base::internal [static], [inherited]`
Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.
Referenced by std::internal().

left

`const fmtflags std::ios_base::left [static], [inherited]`
Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)
Referenced by std::num_put< _CharT, _OutIter >::do_put(), and std::left().

oct

`const fmtflags std::ios_base::oct [static], [inherited]`
Converts integer input or generates integer output in octal base.
Referenced by std::oct().

out

`const openmode std::ios_base::out [static], [inherited]`
Open for output. Default for `ofstream` and fstream.
Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_stringbuf< _CharT, _Traits, std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

right

`const fmtflags std::ios_base::right [static], [inherited]`

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)
Referenced by std::right().

scientific

```
const fmtflags std::ios_base::scientific  [static], [inherited]
```

Generates floating-point output in scientific notation.
Referenced by std::hexfloat(), and std::scientific().

showbase

```
const fmtflags std::ios_base::showbase  [static], [inherited]
```

Generates a prefix indicating the numeric base of generated integer output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::noshowbase(), and std::showbase().

showpoint

```
const fmtflags std::ios_base::showpoint  [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.
Referenced by std::noshowpoint(), and std::showpoint().

showpos

```
const fmtflags std::ios_base::showpos  [static], [inherited]
```

Generates a + sign in non-negative generated numeric output.
Referenced by std::noshowpos(), and std::showpos().

skipws

```
const fmtflags std::ios_base::skipws  [static], [inherited]
```

Skips leading white space before certain input operations.
Referenced by std::noskipws(), std::__detail::operator>>(), and std::skipws().

trunc

```
const openmode std::ios_base::trunc  [static], [inherited]
```

Truncate an existing stream when opening. Default for `ofstream`.

unitbuf

```
const fmtflags std::ios_base::unitbuf  [static], [inherited]
```

Flushes output after each output operation.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::nounitbuf(), and std::unitbuf().

uppercase

```
const fmtflags std::ios_base::uppercase  [static], [inherited]
```

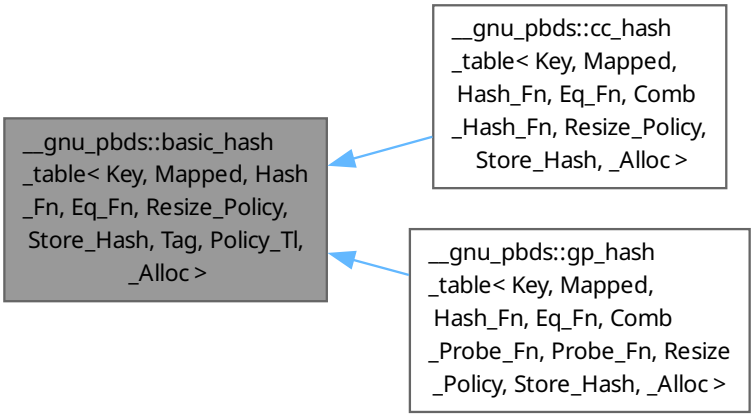
Replaces certain lowercase letters with their uppercase equivalents in generated output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::nouppercase(), and std::uppercase().
The documentation for this class was generated from the following file:

- fstream

5.219 __gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc > Class Template Reference

`#include <assoc_container.hpp>`

Inheritance diagram for __gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc >:



Protected Member Functions

- **basic_hash_table** (const basic_hash_table &other)
- template<typename T0>
  basic_hash_table (T0 t0)
- template<typename T0, typename T1>
  basic_hash_table (T0 t0, T1 t1)
- template<typename T0, typename T1, typename T2>
  basic_hash_table (T0 t0, T1 t1, T2 t2)
- template<typename T0, typename T1, typename T2, typename T3>
  basic_hash_table (T0 t0, T1 t1, T2 t2, T3 t3)
- template<typename T0, typename T1, typename T2, typename T3, typename T4>
  basic_hash_table (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4)
- template<typename T0, typename T1, typename T2, typename T3, typename T4, typename T5>
  basic_hash_table (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5)
- template<typename T0, typename T1, typename T2, typename T3, typename T4, typename T5, typename T6>
  basic_hash_table (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6)
- template<typename T0, typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7>
  basic_hash_table (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6, T7 t7)
- template<typename T0, typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8>
  basic_hash_table (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6, T7 t7, T8 t8)

5.219.1 Detailed Description

template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename Resize_Policy, bool Store_Hash, typename Tag, typename Policy_Tl, typename _Alloc>
class __gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc >

A hashed container abstraction.

Template Parameters

Key	Key type.
Mapped	Map type.
Hash_Fn	Hashing functor.
Eq_Fn	Equal functor.
Resize_Policy	Resizes hash.
Store_Hash	Indicates whether the hash value will be stored along with each key.
Tag	Instantiating data structure type, see container_tag.
Policy_TL	Policy typelist.
_Alloc	Allocator type.

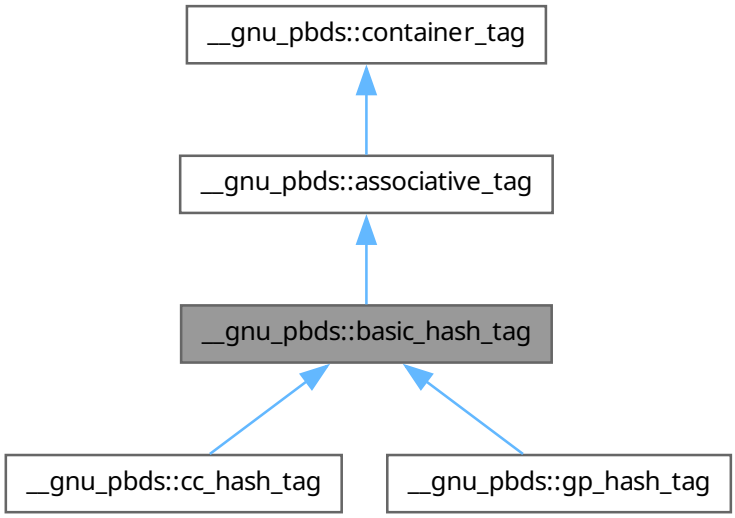
Base is dispatched at compile time via Tag, from the following choices: cc_hash_tag, gp_hash_tag, and descendants of basic_hash_tag.
Base choices are: detail::cc_ht_map, detail::gp_ht_map
The documentation for this class was generated from the following file:

- assoc_container.hpp

5.220 __gnu_pbds::basic_hash_tag Struct Reference

`#include <tag_and_trait.hpp>`

Inheritance diagram for __gnu_pbds::basic_hash_tag:



5.220.1 Detailed Description

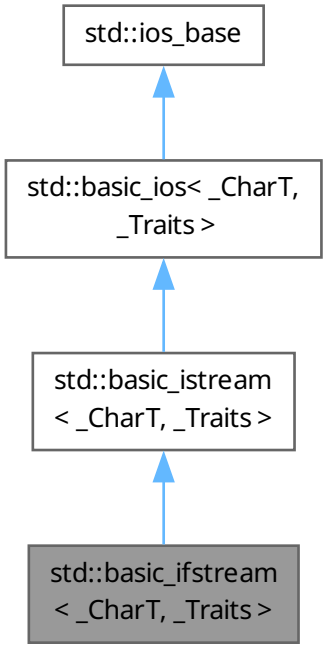
Basic hash structure.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

5.221 std::basic_ifstream< _CharT, _Traits > Class Template Reference

`#include <fstream>`

Inheritance diagram for std::basic_ifstream< _CharT, _Traits >:



Public Types

- typedef ctype< _CharT > **__ctype_type**
- typedef basic_filebuf< char_type, traits_type > **__filebuf_type**
- typedef basic_ios< _CharT, _Traits > **__ios_type**
- typedef basic_istream< char_type, traits_type > **__istream_type**
- typedef num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > **__num_get_type**
- typedef basic_streambuf< _CharT, _Traits > **__streambuf_type**
- typedef _CharT **char_type**
- enum event { **erase_event** , **imbue_event** , **copyfmt_event** }
- typedef void(* event_callback) (event __e, ios_base &__b, int __i)
- typedef traits_type::int_type **int_type**
- typedef _Ios_Iostate iostate
- typedef traits_type::off_type **off_type**
- typedef _Ios_Openmode openmode
- typedef traits_type::pos_type **pos_type**
- typedef _Ios_Seekdir seekdir
- typedef _Traits **traits_type**

- typedef num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > __num_put_type

Public Member Functions

- basic_ifstream ()
- **basic_ifstream** (basic_ifstream &&__rhs)
- template<typename _Path, typename _Require = _If_fs_path<_Path>>
  basic_ifstream (const _Path &__s, ios_base::openmode __mode=ios_base::in)
- **basic_ifstream** (const basic_ifstream &)=delete
- basic_ifstream (const char *__s, ios_base::openmode __mode=ios_base::in)
- basic_ifstream (const std::string &__s, ios_base::openmode __mode=ios_base::in)
- ~basic_ifstream ()
- template<typename _ValueT>
  basic_istream< _CharT, _Traits > & **_M_extract** (_ValueT &__v)
- const locale & _M_getloc () const
- void **_M_setstate** (iostate __state)
- bool bad () const
- void clear (iostate __state=goodbit)
- void close ()
- basic_ios & copyfmt (const basic_ios &__rhs)
- bool eof () const
- iostate exceptions () const
- void exceptions (iostate __except)
- bool fail () const
- char_type fill () const
- char_type fill (char_type __ch)
- fmtflags flags () const
- fmtflags flags (fmtflags __fmtfl)
- streamsize gcount () const
- basic_istream< char > & getline (char_type *__s, streamsize __n, char_type __delim)
- basic_istream< wchar_t > & **getline** (char_type *__s, streamsize __n, char_type __delim)
- locale getloc () const
- bool good () const
- basic_istream< char > & **ignore** (streamsize __n)
- basic_istream< wchar_t > & **ignore** (streamsize __n)
- basic_istream< char > & **ignore** (streamsize __n, int_type __delim)
- basic_istream< wchar_t > & **ignore** (streamsize __n, int_type __delim)
- locale imbue (const locale &__loc)
- bool is_open ()
- bool **is_open** () const
- long & iword (int __ix)
- char narrow (char_type __c, char __dfault) const
- template<typename _Path>
  _If_fs_path< _Path, void > open (const _Path &__s, ios_base::openmode __mode=ios_base::in)
- void open (const char *__s, ios_base::openmode __mode=ios_base::in)
- void open (const std::string &__s, ios_base::openmode __mode=ios_base::in)
- basic_ifstream & **operator=** (basic_ifstream &&__rhs)
- basic_ifstream & **operator=** (const basic_ifstream &)=delete
- __istream_type & operator>> (__streambuf_type *__sb)
- __istream_type & operator>> (void *&__p)
- streamsize precision () const
- streamsize precision (streamsize __prec)
- void *& pword (int __ix)

- __filebuf_type * rdbuf () const
- basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > *__sb)
- iostate rdstate () const
- void register_callback (event_callback __fn, int __index)
- fmtflags setf (fmtflags __fmtfl)
- fmtflags setf (fmtflags __fmtfl, fmtflags __mask)
- void setstate (iostate __state)
- void **swap** (basic_ifstream &__rhs)
- basic_ostream< _CharT, _Traits > * tie () const
- basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > *__tiestr)
- void unsetf (fmtflags __mask)
- char_type widen (char __c) const
- streamsize width () const
- streamsize width (streamsize __wide)

- __istream_type & operator>> (__istream_type &(*__pf)(__istream_type &))
- __istream_type & operator>> (__ios_type &(*__pf)(__ios_type &))
- __istream_type & operator>> (ios_base &(*__pf)(ios_base &))

Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type std::basic_istream::sentry with the second argument (noskipws) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, ios_base::badbit will be turned on in the stream's error state (without causing an ios_base::failure to be thrown) and the original exception will be rethrown if badbit is set in the exceptions mask.

- __istream_type & operator>> (bool &__n)
- __istream_type & operator>> (short &__n)
- __istream_type & operator>> (unsigned short &__n)
- __istream_type & operator>> (int &__n)
- __istream_type & operator>> (unsigned int &__n)
- __istream_type & operator>> (long &__n)
- __istream_type & operator>> (unsigned long &__n)
- __istream_type & operator>> (long long &__n)
- __istream_type & operator>> (unsigned long long &__n)

- __istream_type & operator>> (float &__f)
- __istream_type & operator>> (double &__f)
- __istream_type & operator>> (long double &__f)

Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type std::basic_istream::sentry with the second argument (noskipws) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by gcount().

If an exception is thrown during extraction, ios_base::badbit will be turned on in the stream's error state (without causing an ios_base::failure to be thrown) and the original exception will be rethrown if badbit is set in the exceptions mask.

- int_type get ()
- __istream_type & get (char_type &__c)
- __istream_type & get (char_type *__s, streamsize __n, char_type __delim)
- __istream_type & get (char_type *__s, streamsize __n)
- __istream_type & get (__streambuf_type &__sb, char_type __delim)
- __istream_type & get (__streambuf_type &__sb)
- __istream_type & getline (char_type *__s, streamsize __n, char_type __delim)
- __istream_type & getline (char_type *__s, streamsize __n)
- __istream_type & ignore (streamsize __n, int_type __delim)
- __istream_type & ignore (streamsize __n)
- __istream_type & ignore ()
- int_type peek ()
- __istream_type & read (char_type *__s, streamsize __n)
- streamsize readsome (char_type *__s, streamsize __n)
- __istream_type & putback (char_type __c)
- __istream_type & unget ()
- int sync ()
- pos_type tellg ()
- __istream_type & seekg (pos_type)
- __istream_type & seekg (off_type, ios_base::seekdir)

- operator bool () const
- bool operator! () const

Static Public Member Functions

- static bool sync_with_stdio (bool __sync=true)
- static int xalloc () throw ()

Public Attributes

- class __attribute((__abi_tag__("cxx11"))) failure typedef _Ios_Fmtflags fmtflags

Static Public Attributes

- static const openmode **__noreplace**
- static const fmtflags adjustfield
- static const openmode app
- static const openmode ate
- static const iostate badbit
- static const fmtflags basefield
- static const seekdir beg
- static const openmode binary
- static const fmtflags boolalpha
- static const seekdir cur
- static const fmtflags dec
- static const seekdir end
- static const iostate eofbit
- static const iostate failbit
- static const fmtflags fixed
- static const fmtflags floatfield
- static const iostate goodbit
- static const fmtflags hex
- static const openmode in

- static const fmtflags internal
- static const fmtflags left
- static const fmtflags oct
- static const openmode out
- static const fmtflags right
- static const fmtflags scientific
- static const fmtflags showbase
- static const fmtflags showpoint
- static const fmtflags showpos
- static const fmtflags skipws
- static const openmode trunc
- static const fmtflags unitbuf
- static const fmtflags uppercase

Protected Types

- enum { **_S_local_word_size** }

Protected Member Functions

- void **_M_cache_locale** (const locale &__loc)
- void **_M_call_callbacks** (event __ev) throw ()
- void **_M_dispose_callbacks** (void) throw ()
- template<typename _ValueT>
  __istream_type & **_M_extract** (_ValueT &__v)
- _Words & **_M_grow_words** (int __index, bool __iword)
- void **_M_init** () throw ()
- void **_M_move** (ios_base &) noexcept
- void **_M_swap** (ios_base &__rhs) noexcept
- void init (basic_streambuf< _CharT, _Traits > *__sb)
- void **move** (basic_ios &&__rhs)
- void **move** (basic_ios &__rhs)
- void **set_rdbuf** (basic_streambuf< _CharT, _Traits > *__sb)
- void **swap** (basic_ios &__rhs) noexcept
- void **swap** (basic_istream &__rhs)

Protected Attributes

- _Callback_list * **_M_callbacks**
- const __ctype_type * **_M_ctype**
- iostate **_M_exception**
- char_type **_M_fill**
- bool **_M_fill_init**
- fmtflags **_M_flags**
- streamsize _M_gcount
- locale **_M_ios_locale**
- _Words **_M_local_word** [_S_local_word_size]
- const __num_get_type * **_M_num_get**
- const __num_put_type * **_M_num_put**
- streamsize **_M_precision**
- basic_streambuf< _CharT, _Traits > * **_M_streambuf**
- iostate **_M_streambuf_state**

- basic_ostream< _CharT, _Traits > * **_M_tie**
- streamsize **_M_width**
- _Words * **_M_word**
- int **_M_word_size**
- _Words **_M_word_zero**

5.221.1 Detailed Description

template<typename _CharT, typename _Traits>
class std::basic_ifstream< _CharT, _Traits >

Controlling input for files.

Template Parameters

_CharT	Type of character stream.
_Traits	Traits for character type, defaults to char_traits<_CharT>.

This class supports reading from named files, using the inherited functions from std::basic_istream. To control the associated sequence, an instance of std::basic_filebuf is used, which this page refers to as `sb`.

5.221.2 Member Typedef Documentation

__num_put_type

```
template<typename _CharT, typename _Traits>
typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios< _CharT, _Traits
>::__num_put_type  [inherited]
```

These are non-standard types.

event_callback

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i)  [inherited]
```

The type of an event callback function.

Parameters

_e	One of the members of the event enum.
_b	Reference to the ios_base object.
_i	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several ios_base and basic_ios functions, specifically imbue(), copyfmt(), and ~ios().

iostate

```
typedef _Ios_Iostate std::ios_base::iostate  [inherited]
```

This is a bitmask type.
_Ios_Iostate is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit

- eofbit
- failbit
- goodbit

openmode

```
typedef _Ios_Openmode std::ios_base::openmode  [inherited]
```

This is a bitmask type.
_Ios_Openmode is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type openmode are:

- app
- ate
- binary
- in
- out
- trunc

seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir  [inherited]
```

This is an enumerated type.
_Ios_Seekdir is implementation-defined. Defined values of type seekdir are:

- beg
- cur, equivalent to SEEK_CUR in the C standard library.
- end, equivalent to SEEK_END in the C standard library.

5.221.3 Member Enumeration Documentation

event

```
enum std::ios_base::event  [inherited]
```

The set of events that may be passed to an event callback.
erase_event is used during ~ios() and copyfmt(). imbue_event is used during imbue(). copyfmt_event is used during copyfmt().

5.221.4 Constructor & Destructor Documentation

basic_ifstream() [1/4]

```
template<typename _CharT, typename _Traits>
std::basic_ifstream< _CharT, _Traits >::basic_ifstream ()  [inline]
```

Default constructor.
Initializes sb using its default constructor, and passes &sb to the base class initializer. Does not open any files (you haven't given it a filename to open).

basic_ifstream() [2/4]

```
template<typename _CharT, typename _Traits>
std::basic_ifstream< _CharT, _Traits >::basic_ifstream (
            const char * __s,
            ios_base::openmode __mode = ios_base::in)  [inline], [explicit]
```

Create an input file stream.

Parameters

__s	Null terminated string specifying the filename.
__mode	Open file in specified mode (see std::ios_base).

`ios_base::in` is automatically included in *__mode*.

basic_ifstream() [3/4]

```
template<typename _CharT, typename _Traits>
std::basic_ifstream< _CharT, _Traits >::basic_ifstream (
            const std::string & __s,
            ios_base::openmode __mode = ios_base::in)  [inline], [explicit]
```

Create an input file stream.

Parameters

__s	std::string specifying the filename.
__mode	Open file in specified mode (see std::ios_base).

`ios_base::in` is automatically included in *__mode*.

basic_ifstream() [4/4]

```
template<typename _CharT, typename _Traits>
template<typename _Path, typename _Require = _If_fs_path<_Path>>
std::basic_ifstream< _CharT, _Traits >::basic_ifstream (
            const _Path & __s,
            ios_base::openmode __mode = ios_base::in)  [inline]
```

Create an input file stream.

Parameters

__s	filesystem::path specifying the filename.
__mode	Open file in specified mode (see std::ios_base).

`ios_base::in` is automatically included in *__mode*.

~basic_ifstream()

```
template<typename _CharT, typename _Traits>
std::basic_ifstream< _CharT, _Traits >::~basic_ifstream ()  [inline]
```

The destructor does nothing.
The file is closed by the filebuf object, not the formatting stream.

5.221.5 Member Function Documentation

_M_getloc()

```
const locale & std::ios_base::_M_getloc () const  [inline], [inherited]
```

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.
Referenced by std::money_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter
std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIte
std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::num_put< _CharT, _OutIter
std::time_put< _CharT, _OutIter >::do_put(), std::time_get< _CharT, _InIter >::get(), and std::time_put< _CharT, _OutIter >::put().

bad()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry().

clear()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::clear (
             iostate __state = goodbit)  [inherited]
```

[Re]sets the error state.

Parameters

__state	The new state flag(s) to set.

See std::ios_base::iostate for the possible bit values. Most users will not need to pass an argument.
References std::ios_base::badbit, exceptions(), rdbuf(), and rdstate().
Referenced by std::basic_ios< char_type, traits_type >::exceptions(), std::__detail::operator>>(), std::basic_istream< _CharT, _Traits >
rdbuf(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char_type, traits
and std::basic_istream< _CharT, _Traits >::unget().

close()

```
template<typename _CharT, typename _Traits>
void std::basic_ifstream< _CharT, _Traits >::close ()  [inline]
```

Close the file.
Calls `std::basic_filebuf::close()`. If that function fails, `failbit` is set in the stream's error state.

copyfmt()

```
template<typename _CharT, typename _Traits>
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
            const basic_ios< _CharT, _Traits > & __rhs)  [inherited]
```

Copies fields of __rhs into this.

Parameters

__rhs	The source values for the copies.

Returns

Reference to this object.

All fields of __rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase_event. After copying, each (new) callback is invoked with copyfmt_event. The final step is to copy exceptions().
References basic_ios(), std::__addressof(), exceptions(), fill(), std::ios_base::flags(), std::ios_base::getloc(), std::ios_base::precision(), tie(), std::tie(), and std::ios_base::width().

eof()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

exceptions() [1/2]

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions () const  [inline], [nodiscard], [inherited]
```

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.
Referenced by clear(), and copyfmt().

exceptions() [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
            iostate __except)  [inline], [inherited]
```

Throwing exceptions on errors.

Parameters

__except	The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type std::ios_base::failure is thrown.
If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
```

```
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr « "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr « "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.
Referenced by std::basic_ios< char_type, traits_type >::operator bool(), std::basic_ios< char_type, traits_type >::operator!(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >
std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::
and std::regex_traits< _Ch_type >::value().

fill() [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const  [inline], [nodiscard], [inherited]
```

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.
Referenced by copyfmt().

fill() [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
             char_type __ch)  [inline], [inherited]
```

Sets a new *empty* character.

Parameters

__ch	The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

flags() [1/2]

```
fmtflags std::ios_base::flags () const  [inline], [nodiscard], [inherited]
```
Access to format flags.

Returns

The format control flags for both input and output.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >
std::num_put< _CharT, _OutIter >::do_put(), std::num_put< _CharT, _OutIter >::do_put(), std::chrono::operator<<(),
std::operator<<(), std::__detail::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(),
std::operator>>(), std::operator>>(), and std::operator>>().

flags() [2/2]

```
fmtflags std::ios_base::flags (
            fmtflags __fmtfl)  [inline], [inherited]
```
Setting new format flags all at once.

Parameters

__fmtfl	The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *__fmtfl*.
References fmtflags.

gcount()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::gcount () const  [inline], [inherited]
```
Character counting.

Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

get() [1/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::get (
            void )  [inherited]
```
Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().
References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit,
std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

get() [2/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::get (
            __streambuf_type & __sb)  [inline], [inherited]
```

Extraction into another streambuf.

Parameters

__sb	A streambuf in which to store data.

Returns

*this

Returns `get`(__sb,widen('\n')).

get() [3/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
            __streambuf_type & __sb,
            char_type __delim)  [inherited]
```

Extraction into another streambuf.

Parameters

__sb	A streambuf in which to store data.
__delim	A "stop" character.

Returns

*this

Characters are extracted and inserted into *__sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals *__delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.
References _M_gcount, and std::ios_base::goodbit.

get() [4/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
            char_type & __c)  [inherited]
```

Simple extraction.

Parameters

_↩ _c	The character in which to store data.

Returns

*this

Tries to extract a character and store it in __c. If none are available, sets failbit and returns traits::eof().

Note

This function is not overloaded on signed char and unsigned char.

References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

get() [5/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::get (
            char_type * __s,
            streamsize __n)  [inline], [inherited]
```

Simple multiple-character extraction.

Parameters

_↩ _s	Pointer to an array.
_↩ _n	Maximum number of characters to store in *s*.

Returns

*this

Returns `get(__s,__n,widen('\n'))`.

get() [6/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
            char_type * __s,
            streamsize __n,
            char_type __delim)  [inherited]
```

Simple multiple-character extraction.

Parameters

__s	Pointer to an array.
__n	Maximum number of characters to store in __s.
__delim	A "stop" character.

Returns

*this

Characters are extracted and stored into __*s* until one of the following happens:

- __n-1 characters are stored
- the input sequence reaches EOF
- the next character equals __*delim*, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.
In any case, a null character is stored into the next location in the array.

Note

This function is not overloaded on signed char and unsigned char.

References _M_gcount, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_streambuf< _CharT, _Traits >:

getline() [1/3]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::getline (
            char_type * __s,
            streamsize __n)  [inline], [inherited]
```

String extraction.

Parameters

_↩ _*s*	A character array in which to store the data.
_↩ _*n*	Maximum number of characters to extract.

Returns

*this

Returns getline(__s,__n,widen('\n')).

getline() [2/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::getline (
            char_type * __s,
            streamsize __n,
            char_type __delim)  [inherited]
```

String extraction.

Parameters

__*s*	A character array in which to store the data.
__*n*	Maximum number of characters to extract.
__*delim*	A "stop" character.

Returns

*this

Extracts and stores characters into *__s* until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the *__s* array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)
In any case, a null character is stored in the next location in the array.
References _M_gcount, std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_streambuf< _CharT, _Traits >::sbumpc(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _Cha

getline() [3/3]

```
basic_istream< char > & std::basic_istream< char >::getline (
            char_type * __s,
            streamsize __n,
            char_type __delim)  [inherited]
```

Explicit specialization declarations, defined in src/istream.cc.

getloc()

```
locale std::ios_base::getloc () const  [inline], [nodiscard], [inherited]
```

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::money_put< _CharT, _OutIter >::do_put(), std::basic_ios< _CharT, _Tr std::chrono::operator<<(), std::operator>>(), std::operator>>(), and std::ws().

good()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if no error flags are set.

A wrapper around rdstate.
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::__detail::operator>>().

ignore() [1/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
            void )  [inherited]
```

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().
References _M_gcount, std::ios_base::goodbit, and std::basic_ios< _CharT, _Traits >::rdbuf().

ignore() [2/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
            streamsize __n)  [inherited]
```

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().
References _M_gcount, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_streambuf< _CharT, _Traits >:

ignore() [3/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
            streamsize __n,
            int_type __delim)  [inherited]
```

Discarding characters.

Parameters

__n	Number of characters to discard.
__delim	A "stop" character.

Returns

*this

Extracts characters and throws them away until one of the following happens:

- if *__n* != `std::numeric_limits<int>::max()`, *__n* characters are extracted
- the input sequence reaches end-of-file
- the next character equals *__delim* (in this case, the character is extracted); note that this condition will never occur if *__delim* equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.
References _M_gcount, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_streambuf< _CharT, _Traits >:

imbue()

```
template<typename _CharT, typename _Traits>
locale std::basic_ios< _CharT, _Traits >::imbue (
            const locale & __loc)  [inherited]
```

Moves to a new locale.

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.
Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html
References std::ios_base::getloc(), std::ios_base::imbue(), and rdbuf().
Referenced by std::chrono::operator<<().

init()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::init (
            basic_streambuf< _CharT, _Traits > * __sb)  [protected], [inherited]
```

All setup is performed here.
This is called from the public constructor. It is not virtual and cannot be redefined.
Referenced by std::basic_ios< char_type, traits_type >::basic_ios(), and std::basic_ostream< char_type, traits_type >::basic_ostream().

is_open()

```
template<typename _CharT, typename _Traits>
bool std::basic_ifstream< _CharT, _Traits >::is_open ()  [inline], [nodiscard]
```

Wrapper to test for an open file.

Returns

`rdbuf()->is_open()`

iword()

```
long & std::ios_base::iword (
            int __ix)  [inline], [inherited]
```

Access to integer array.

Parameters

__ix	Index into the array.

Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.
References iword().
Referenced by iword().

narrow()

```
template<typename _CharT, typename _Traits>
char std::basic_ios< _CharT, _Traits >::narrow (
            char_type __c,
            char __dfault) const  [inline], [inherited]
```

Squeezes characters.

Parameters

__c	The character to narrow.
__dfault	The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.
Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c,dfault)
```

Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html

open() [1/3]

```
template<typename _CharT, typename _Traits>
template<typename _Path>
_If_fs_path< _Path, void > std::basic_ifstream< _CharT, _Traits >::open (
            const _Path & __s,
            ios_base::openmode __mode = ios_base::in)  [inline]
```

Opens an external file.

Parameters

__s	The name of the file, as a filesystem::path.
__mode	The open mode flags.

Calls `std::basic_filebuf::open(__s,__mode|in)`. If that function fails, `failbit` is set in the stream's error state.

open() [2/3]

```
template<typename _CharT, typename _Traits>
void std::basic_ifstream< _CharT, _Traits >::open (
            const char * __s,
            ios_base::openmode __mode = ios_base::in)  [inline]
```

Opens an external file.

Parameters

__s	The name of the file.
__mode	The open mode flags.

Calls `std::basic_filebuf::open(s,__mode|in)`. If that function fails, `failbit` is set in the stream's error state.

open() [3/3]

```
template<typename _CharT, typename _Traits>
void std::basic_ifstream< _CharT, _Traits >::open (
            const std::string & __s,
            ios_base::openmode __mode = ios_base::in)  [inline]
```

Opens an external file.

Parameters

__s	The name of the file.
__mode	The open mode flags.

Calls `std::basic_filebuf::open(__s,__mode|in)`. If that function fails, `failbit` is set in the stream's error state.

operator bool()

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool () const  [inline], [explicit], [nodiscard],
[inherited]
```

The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

operator"!()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator!  () const  [inline], [nodiscard], [inherited]
```

The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

operator>>() [1/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            __ios_type &(* __pf )(__ios_type &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the iomanip header.

operator>>() [2/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            __istream_type &(* __pf )(__istream_type &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the iomanip header.

operator>>() [3/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
            __streambuf_type * __sb)  [inherited]
```

Extracting into another streambuf.

Parameters

__sb	A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.
If __sb is NULL, the stream will set failbit in its error state.
Characters are extracted from this stream and inserted into the __sb streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.
References std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

operator>>() [4/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            bool & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [5/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            double & __f)  [inline], [inherited]
```

Floating point arithmetic extractors.

Parameters

↩ _↩ ↩ _↩ f	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [6/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            float & __f)  [inline], [inherited]
```

Floating point arithmetic extractors.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

operator>>() [7/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
            int & __n)  [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ *_n*	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.
References std::ios_base::badbit, std::ios_base::failbit, std::num_get< _CharT, _InIter >::get(), std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::setstate(), and std::use_facet().

operator>>() [8/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            ios_base &(* __pf )(ios_base &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the iomanip header.

operator>>() [9/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            long & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [10/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            long double & __f)  [inline], [inherited]
```

Floating point arithmetic extractors.

Parameters

↩ _↩ ↩ _↩ f	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [11/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            long long & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [12/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
            short & __n)  [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.
References std::ios_base::badbit, std::ios_base::failbit, std::num_get< _CharT, _InIter >::get(), std::ios_base::goodbit, and std::use_facet().

operator>>() [13/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned int & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [14/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned long & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [15/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned long long & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [16/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned short & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [17/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            void *& __p)  [inline], [inherited]
```

Basic arithmetic extractors.

Parameters

_↩ _p	A variable of pointer type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

peek()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::peek (
            void )  [inherited]
```

Looking ahead in the stream.

Returns

The next character, or eof().

If, after constructing the sentry object, good() is false, returns traits::eof(). Otherwise reads but does not extract the next input character.

precision() [1/2]

```
streamsize std::ios_base::precision () const  [inline], [nodiscard], [inherited]
```

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::chrono::operator<<().

precision() [2/2]

```
streamsize std::ios_base::precision (
            streamsize __prec)  [inline], [inherited]
```

Changing flags.

Parameters

__prec	The new precision value.

Returns

The previous value of precision().

putback()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::putback (
            char_type __c)  [inherited]
```

Unextracting a single character.

Parameters

_↩ _c	The character to push back into the input stream.

Returns

*this

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.
If `rdbuf()` is null or if `sputbackc()` fails, sets badbit in the error state.

Note

This function first clears eofbit. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References _M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::rdstate(), std::basic_ios< _CharT, _Traits >::setstate(), and std::basic_streambuf< _CharT, _Traits >::sputbackc().

pword()

```
void *& std::ios_base::pword (
            int __ix)  [inline], [inherited]
```

Access to void pointer array.

Parameters

_↩ _*ix*	Index into the array.

Returns

A reference to a void* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

rdbuf() [1/2]

```
template<typename _CharT, typename _Traits>
__filebuf_type * std::basic_ifstream< _CharT, _Traits >::rdbuf () const  [inline], [nodiscard]
```

Accessing the underlying buffer.

Returns

The current basic_filebuf buffer.

This hides both signatures of std::basic_ios::rdbuf().

rdbuf() [2/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
            basic_streambuf< _CharT, _Traits > * __sb)  [inherited]
```

Changing the underlying buffer.

Parameters

__sb	The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.
Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream     foo;         // or some other derived type
std::streambuf*  p = .....;

foo.ios::rdbuf(p);            // ios == basic_ios<char>
```

References clear().

rdstate()

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::rdstate () const  [inline], [nodiscard], [inherited]
```

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See std::ios_base::iostate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().
Referenced by std::basic_ios< char_type, traits_type >::bad(), clear(), std::basic_ios< char_type, traits_type >::eof(), std::basic_ios< char_type, traits_type >::fail(), std::basic_ios< char_type, traits_type >::good(), std::basic_istream< _CharT, _Traits >::
std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char_type, traits_type >
and std::basic_istream< _CharT, _Traits >::unget().

read()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::read (
            char_type * __s,
            streamsize __n)  [inherited]
```

Extraction without delimiters.

Parameters

_↩ _s	A character array.
_↩ _n	Maximum number of characters to store.

Returns

*this

If the stream state is `good()`, extracts characters and stores them into *__s* until one of the following happens:

- *__n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

Note

This function is not overloaded on signed char and unsigned char.

References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

readsome()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::readsome (
            char_type * __s,
            streamsize __n)  [inherited]
```

Extraction until the buffer is exhausted, but no more.

Parameters

_↩ _s	A character array.
_↩ _n	Maximum number of characters to store.

Returns

The number of characters extracted.

Extracts characters and stores them into __*s* depending on the number of characters remaining in the streambuf's buffer, `rdbuf()->in_avail()`, called `A` here:

- if `A == -1`, sets eofbit and extracts no characters
- if `A == 0`, extracts no characters
- if `A > 0`, extracts `min(A,n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.
References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::min(), std::basic_ios< _CharT, _Traits >::
and std::basic_ios< _CharT, _Traits >::setstate().

register_callback()

```
void std::ios_base::register_callback (
            event_callback __fn,
            int __index)  [inherited]
```

Add the callback __fn with parameter __index.

Parameters

__fn	The function to add.
__index	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.
References fmtflags.

seekg() [1/2]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
            off_type __off,
            ios_base::seekdir __dir)  [inherited]
```

Changing the current read position.

Parameters

__off	A file offset object.
__dir	The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(__off,__dir)`. If that function fails, sets failbit.

Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::basic_ios< _CharT, _Traits >::fail(
std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::in, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits
and std::basic_ios< _CharT, _Traits >::setstate().

seekg() [2/2]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
            pos_type __pos)  [inherited]
```

Changing the current read position.

Parameters

__pos	A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets failbit.

Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::basic_ios< _CharT, _Traits >::fail(
std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::in, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits
and std::basic_ios< _CharT, _Traits >::setstate().

setf() [1/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl)  [inline], [inherited]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.
References fmtflags.
Referenced by std::__detail::operator>>().

setf() [2/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl,
            fmtflags __mask)  [inline], [inherited]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.
__mask	The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.
References fmtflags.

setstate()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
            iostate __state)  [inline], [inherited]
```

Sets additional flags in the error state.

Parameters

__state	The additional state flag(s) to set.

See std::ios_base::iostate for the possible bit values.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::getline(), std::getline(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _Cha std::operator>>(), std::operator>>(), std::operator>>(), std::tr2::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >: std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits > std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), and std::ws().

sync()

```
template<typename _CharT, typename _Traits>
int std::basic_istream< _CharT, _Traits >::sync (
            void )  [inherited]
```

Synchronizing the stream buffer.

Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.
Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit and returns -1.
Otherwise, returns 0.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_streambuf< _CharT, _Traits >::pubsync(), std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

sync_with_stdio()

```
static bool std::ios_base::sync_with_stdio (
            bool __sync = true)  [static], [inherited]
```

Interaction with the standard C I/O objects.

Parameters

__sync	Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary

tellg()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::pos_type std::basic_istream< _CharT, _Traits >::tellg (
            void )  [inherited]
```

Getting the current read position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type`(-1) to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0,cur,in)`.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with putback, unget and seekg, eofbit is not cleared first.

References std::ios_base::badbit, std::ios_base::cur, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::in, and std::basic_ios< _CharT, _Traits >::rdbuf().

tie() [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie () const  [inline],
[nodiscard], [inherited]
```

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and copyfmt().

tie() [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie (
            basic_ostream< _CharT, _Traits > * __tiestr)  [inline], [inherited]
```

Ties this stream to an output stream.

Parameters

__tiestr	The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

unget()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::unget (
            void )  [inherited]
```

Unextracting the previous character.

Returns

*this

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.
If `rdbuf()` is null or if `sungetc()` fails, sets badbit in the error state.

Note

This function first clears eofbit. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References _M_gcount, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::ios_base::goodbit, and std::basic_ios< _CharT, _Traits >::rdstate().
Referenced by std::__detail::operator>>().

unsetf()

```
void std::ios_base::unsetf (
            fmtflags __mask)  [inline], [inherited]
```

Clearing format flags.

Parameters

__mask	The flags to unset.

This function clears *__mask* in the format flags.
References fmtflags.

widen()

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
            char __c) const  [inline], [inherited]
```

Widens characters.

Parameters

_↩ _c	The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.
Returns the result of
`std::use_facet<ctype<char_type> >(getloc()).widen(c)`
Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html
Referenced by std::basic_ios< char_type, traits_type >::fill(), std::getline(), std::getline(), and std::tr2::operator>>().

width() [1/2]

```
streamsize std::ios_base::width () const  [inline], [nodiscard], [inherited]
```
Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_put< _CharT, _OutIter >::do_put(), std::operator>>(), and std::operator>>().

width() [2/2]

```
streamsize std::ios_base::width (
             streamsize __wide)  [inline], [inherited]
```
Changing flags.

Parameters

__wide	The new width value.

Returns

The previous value of width().

xalloc()

```
static int std::ios_base::xalloc () throw ( )  [static], [inherited]
```
Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the iword and pword functions. The expectation is that an application calls xalloc in order to obtain an index in the iword and pword arrays that can be used without fear of conflict.
The implementation maintains a static variable that is incremented and returned on each invocation. xalloc is guaranteed to return an index that is safe to use in the iword and pword arrays.

5.221.6 Member Data Documentation

_M_gcount

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::_M_gcount  [protected], [inherited]
```
The number of characters extracted in the previous unformatted function; see gcount().
Referenced by get(), get(), get(), get(), getline(), ignore(), ignore(), ignore(), putback(), read(), readsome(), and unget().

adjustfield

```
const fmtflags std::ios_base::adjustfield  [static], [inherited]
```
A mask of left|right|internal. Useful for the 2-arg form of `setf`.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::internal(), std::left(), and std::right().

app

```
const openmode std::ios_base::app  [static], [inherited]
```
Seek to end before each write.
Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

ate

```
const openmode std::ios_base::ate  [static], [inherited]
```
Open and seek to end immediately after opening.
Referenced by std::basic_filebuf< _CharT, _Traits >::open().

badbit

```
const iostate std::ios_base::badbit  [static], [inherited]
```
Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< char_type, traits_type >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::basic_ios< _CharT, _Traits >::clear(), std::basic_ostream< _CharT, _Trai std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_ostream< _CharT, _Traits >::opera std::basic_ostream< char_type, traits_type >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >: std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< char_type, traits_ty std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_istream< char_type, traits_type and std::ws().

basefield

```
const fmtflags std::ios_base::basefield  [static], [inherited]
```
A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.
Referenced by std::dec(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::hex(), and std::oct().

beg

```
const seekdir std::ios_base::beg  [static], [inherited]
```
Request a seek relative to the beginning of the stream.
Referenced by std::basic_filebuf< _CharT, _Traits >::seekpos().

binary

```
const openmode std::ios_base::binary  [static], [inherited]
```
Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.↩filestreams.binary.
Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

boolalpha

```
const fmtflags std::ios_base::boolalpha [static], [inherited]
```

Insert/extract `bool` in alphabetic rather than numeric format.

Referenced by std::boolalpha(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::noboolalpha().

cur

```
const seekdir std::ios_base::cur [static], [inherited]
```

Request a seek relative to the current position within the sequence.

Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

dec

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.

Referenced by std::dec().

end

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.

Referenced by std::basic_filebuf< _CharT, _Traits >::open(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff().

eofbit

```
const iostate std::ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >:: std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::ge std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< char_type, traits_type >::g std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< char_type, traits_type >::read(), std::basic_istream< _CharT, _Traits std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< char_type, traits_ty std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

failbit

```
const iostate std::ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::ge std::basic_istream< char_type, traits_type >::get(), std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::oper std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::s std::basic_ostream< _CharT, _Traits >::seekp(), and std::basic_ostream< _CharT, _Traits >::seekp().

fixed

```
const fmtflags std::ios_base::fixed  [static], [inherited]
```
Generate floating-point output in fixed-point notation.
Referenced by std::fixed(), and std::hexfloat().

floatfield

```
const fmtflags std::ios_base::floatfield  [static], [inherited]
```
A mask of scientific|fixed. Useful for the 2-arg form of `setf`.
Referenced by std::defaultfloat(), std::fixed(), std::hexfloat(), and std::scientific().

goodbit

```
const iostate std::ios_base::goodbit  [static], [inherited]
```
Indicates all is well.
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::d std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >: std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore() std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ostream< _CharT, _Traits > std::basic_ostream< char_type, traits_type >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >:: std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits std::basic_istream< char_type, traits_type >::seekg(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Trait and std::ws().

hex

```
const fmtflags std::ios_base::hex  [static], [inherited]
```
Converts integer input or generates integer output in hexadecimal base.
Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::hex().

in

```
const openmode std::ios_base::in  [static], [inherited]
```
Open for input. Default for `ifstream` and fstream.
Referenced by std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_stringbuf< _CharT, _Traits, _ std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_filebuf< _CharT, _Traits >::underflow(), std::basic_stringbuf< _CharT, _Traits, _ and std::basic_filebuf< _CharT, _Traits >::xsgetn().

internal

```
const fmtflags std::ios_base::internal  [static], [inherited]
```
Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.
Referenced by std::internal().

left

```
const fmtflags std::ios_base::left  [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)
Referenced by std::num_put< _CharT, _OutIter >::do_put(), and std::left().

oct

```
const fmtflags std::ios_base::oct  [static], [inherited]
```
Converts integer input or generates integer output in octal base.
Referenced by std::oct().

out

```
const openmode std::ios_base::out  [static], [inherited]
```
Open for output. Default for `ofstream` and fstream.
Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_stringbuf< _CharT, _Traits, std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

right

```
const fmtflags std::ios_base::right  [static], [inherited]
```
Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)
Referenced by std::right().

scientific

```
const fmtflags std::ios_base::scientific  [static], [inherited]
```
Generates floating-point output in scientific notation.
Referenced by std::hexfloat(), and std::scientific().

showbase

```
const fmtflags std::ios_base::showbase  [static], [inherited]
```
Generates a prefix indicating the numeric base of generated integer output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::noshowbase(), and std::showbase().

showpoint

```
const fmtflags std::ios_base::showpoint  [static], [inherited]
```
Generates a decimal-point character unconditionally in generated floating-point output.
Referenced by std::noshowpoint(), and std::showpoint().

showpos

```
const fmtflags std::ios_base::showpos  [static], [inherited]
```
Generates a + sign in non-negative generated numeric output.
Referenced by std::noshowpos(), and std::showpos().

skipws

```
const fmtflags std::ios_base::skipws  [static], [inherited]
```
Skips leading white space before certain input operations.
Referenced by std::noskipws(), std::__detail::operator>>(), and std::skipws().

trunc

```
const openmode std::ios_base::trunc  [static], [inherited]
```
Truncate an existing stream when opening. Default for `ofstream`.

unitbuf

```
const fmtflags std::ios_base::unitbuf  [static], [inherited]
```
Flushes output after each output operation.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::nounitbuf(), and std::unitbuf().

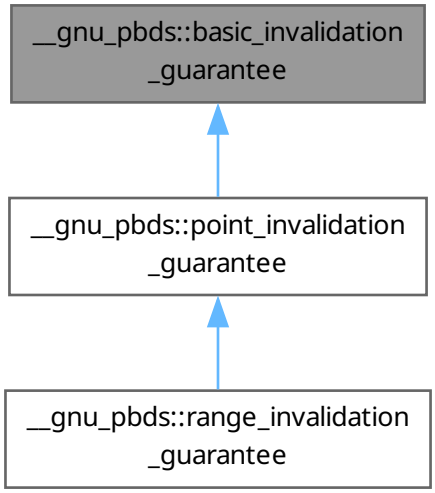
uppercase

```
const fmtflags std::ios_base::uppercase  [static], [inherited]
```
Replaces certain lowercase letters with their uppercase equivalents in generated output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::nouppercase(), and std::uppercase().
The documentation for this class was generated from the following file:

- fstream

5.222 __gnu_pbds::basic_invalidation_guarantee Struct Reference

```
#include <tag_and_trait.hpp>
```
Inheritance diagram for __gnu_pbds::basic_invalidation_guarantee:



5.222.1 Detailed Description

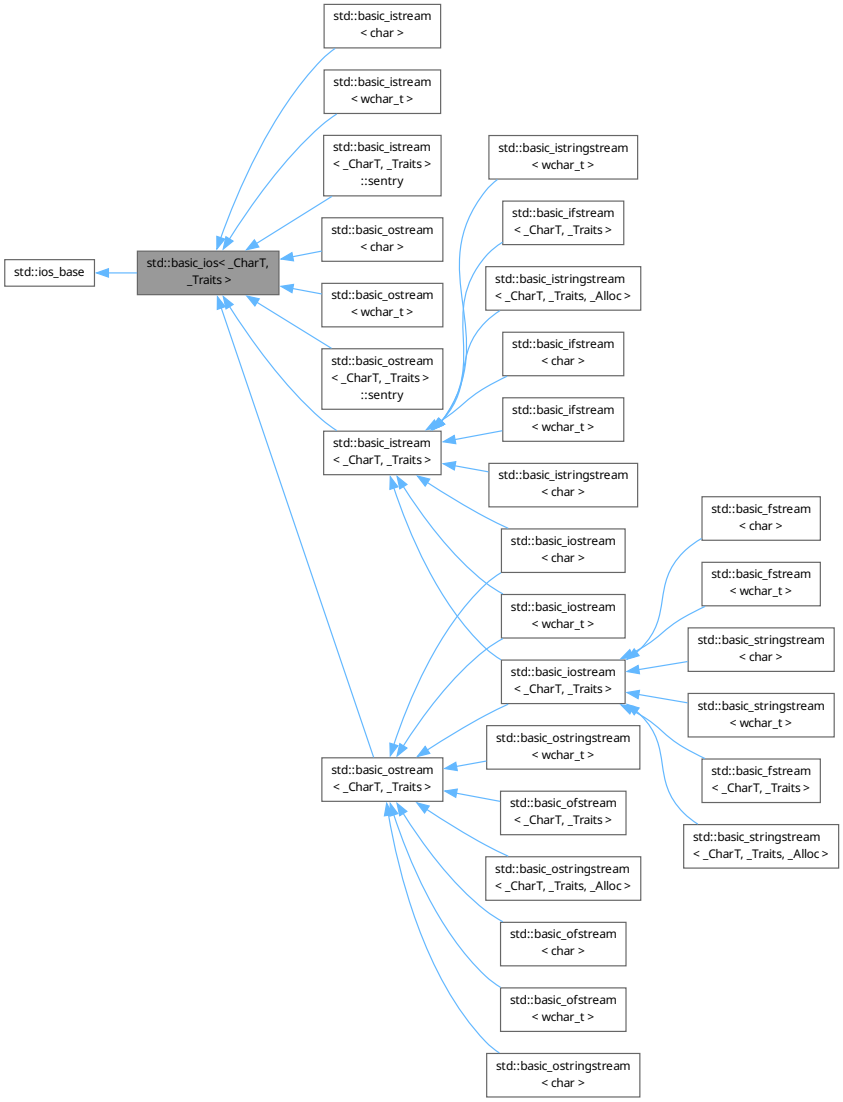
Signifies a basic invalidation guarantee that any iterator, pointer, or reference to a container object's mapped value type is valid as long as the container is not modified.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

5.223 std::basic_ios< _CharT, _Traits > Class Template Reference

`#include <basic_ios.h>`

Inheritance diagram for std::basic_ios< _CharT, _Traits >:



Public Types

- enum event { **erase_event** , **imbue_event** , **copyfmt_event** }
- typedef void(∗ event_callback) (event __e, ios_base &__b, int __i)
- typedef _Ios_Iostate iostate
- typedef _Ios_Openmode openmode
- typedef _Ios_Seekdir seekdir

- typedef _CharT char_type
- typedef _Traits::int_type int_type

- typedef _Traits::pos_type pos_type
- typedef _Traits::off_type off_type
- typedef _Traits traits_type

- typedef ctype< _CharT > __ctype_type
- typedef num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > __num_put_type
- typedef num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > __num_get_type

Public Member Functions

- basic_ios (basic_streambuf< _CharT, _Traits > *__sb)
- virtual ~basic_ios ()
- const locale & _M_getloc () const
- void **_M_setstate** (iostate __state)
- bool bad () const
- void clear (iostate __state=goodbit)
- basic_ios & copyfmt (const basic_ios &__rhs)
- bool eof () const
- iostate exceptions () const
- void exceptions (iostate __except)
- bool fail () const
- char_type fill () const
- char_type fill (char_type __ch)
- fmtflags flags () const
- fmtflags flags (fmtflags __fmtfl)
- locale getloc () const
- bool good () const
- locale imbue (const locale &__loc)
- long & iword (int __ix)
- char narrow (char_type __c, char __dfault) const
- streamsize precision () const
- streamsize precision (streamsize __prec)
- void *& pword (int __ix)
- basic_streambuf< _CharT, _Traits > * rdbuf () const
- basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > *__sb)
- iostate rdstate () const
- void register_callback (event_callback __fn, int __index)
- fmtflags setf (fmtflags __fmtfl)
- fmtflags setf (fmtflags __fmtfl, fmtflags __mask)
- void setstate (iostate __state)
- basic_ostream< _CharT, _Traits > * tie () const
- basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > *__tiestr)
- void unsetf (fmtflags __mask)
- char_type widen (char __c) const
- streamsize width () const
- streamsize width (streamsize __wide)

- operator bool () const
- bool operator! () const

Static Public Member Functions

- static bool sync_with_stdio (bool __sync=true)
- static int xalloc () throw ()

Public Attributes

- class __attribute((__abi_tag__("cxx11"))) failure typedef _Ios_Fmtflags fmtflags

Static Public Attributes

- static const openmode **__noreplace**
- static const fmtflags adjustfield
- static const openmode app
- static const openmode ate
- static const iostate badbit
- static const fmtflags basefield
- static const seekdir beg
- static const openmode binary
- static const fmtflags boolalpha
- static const seekdir cur
- static const fmtflags dec
- static const seekdir end
- static const iostate eofbit
- static const iostate failbit
- static const fmtflags fixed
- static const fmtflags floatfield
- static const iostate goodbit
- static const fmtflags hex
- static const openmode in
- static const fmtflags internal
- static const fmtflags left
- static const fmtflags oct
- static const openmode out
- static const fmtflags right
- static const fmtflags scientific
- static const fmtflags showbase
- static const fmtflags showpoint
- static const fmtflags showpos
- static const fmtflags skipws
- static const openmode trunc
- static const fmtflags unitbuf
- static const fmtflags uppercase

Protected Types

- enum { **_S_local_word_size** }

Protected Member Functions

- basic_ios ()
- **basic_ios** (const basic_ios &)=delete
- void **_M_cache_locale** (const locale &__loc)
- void **_M_call_callbacks** (event __ev) throw ()
- void **_M_dispose_callbacks** (void) throw ()
- _Words & **_M_grow_words** (int __index, bool __iword)
- void **_M_init** () throw ()
- void **_M_move** (ios_base &) noexcept
- void **_M_swap** (ios_base &__rhs) noexcept
- void init (basic_streambuf< _CharT, _Traits > *__sb)
- void **move** (basic_ios &&__rhs)
- void **move** (basic_ios &__rhs)
- basic_ios & **operator=** (const basic_ios &)=delete
- void **set_rdbuf** (basic_streambuf< _CharT, _Traits > *__sb)
- void **swap** (basic_ios &__rhs) noexcept

Protected Attributes

- _Callback_list * **_M_callbacks**
- const __ctype_type * **_M_ctype**
- iostate **_M_exception**
- char_type **_M_fill**
- bool **_M_fill_init**
- fmtflags **_M_flags**
- locale **_M_ios_locale**
- _Words **_M_local_word** [_S_local_word_size]
- const __num_get_type * **_M_num_get**
- const __num_put_type * **_M_num_put**
- streamsize **_M_precision**
- basic_streambuf< _CharT, _Traits > * **_M_streambuf**
- iostate **_M_streambuf_state**
- basic_ostream< _CharT, _Traits > * **_M_tie**
- streamsize **_M_width**
- _Words * **_M_word**
- int **_M_word_size**
- _Words **_M_word_zero**

5.223.1 Detailed Description

template<typename _CharT, typename _Traits>
class std::basic_ios< _CharT, _Traits >

Template class basic_ios, virtual base class for all stream classes.

Template Parameters

_CharT	Type of character stream.
_Traits	Traits for character type, defaults to char_traits<_CharT>.

Most of the member functions called dispatched on stream objects (e.g., `std::cout.foo(bar);`) are consolidated in this class.

5.223.2 Member Typedef Documentation

__ctype_type

```
template<typename _CharT, typename _Traits>
typedef ctype<_CharT> std::basic_ios< _CharT, _Traits >::__ctype_type
```

These are non-standard types.

__num_get_type

```
template<typename _CharT, typename _Traits>
typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits> > std::basic_ios< _CharT, _Traits
>::__num_get_type
```

These are non-standard types.

__num_put_type

```
template<typename _CharT, typename _Traits>
typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios< _CharT, _Traits
>::__num_put_type
```

These are non-standard types.

char_type

```
template<typename _CharT, typename _Traits>
typedef _CharT std::basic_ios< _CharT, _Traits >::char_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

event_callback

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i)  [inherited]
```

The type of an event callback function.

Parameters

_e	One of the members of the event enum.
_b	Reference to the ios_base object.
_i	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several ios_base and basic_ios functions, specifically imbue(), copyfmt(), and ~ios().

int_type

```
template<typename _CharT, typename _Traits>
typedef _Traits::int_type std::basic_ios< _CharT, _Traits >::int_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

iostate

```
typedef _Ios_Iostate std::ios_base::iostate  [inherited]
```

This is a bitmask type.
_Ios_Iostate is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit
- eofbit
- failbit
- goodbit

off_type

```
template<typename _CharT, typename _Traits>
typedef _Traits::off_type std::basic_ios< _CharT, _Traits >::off_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

openmode

```
typedef _Ios_Openmode std::ios_base::openmode  [inherited]
```

This is a bitmask type.
_Ios_Openmode is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type openmode are:

- app
- ate
- binary
- in
- out
- trunc

pos_type

```
template<typename _CharT, typename _Traits>
typedef _Traits::pos_type std::basic_ios< _CharT, _Traits >::pos_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir  [inherited]
```

This is an enumerated type.
_Ios_Seekdir is implementation-defined. Defined values of type seekdir are:

- beg
- cur, equivalent to SEEK_CUR in the C standard library.
- end, equivalent to SEEK_END in the C standard library.

traits_type

```
template<typename _CharT, typename _Traits>
typedef _Traits std::basic_ios< _CharT, _Traits >::traits_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

5.223.3 Member Enumeration Documentation

event

```
enum std::ios_base::event  [inherited]
```

The set of events that may be passed to an event callback.
erase_event is used during ~ios() and copyfmt(). imbue_event is used during imbue(). copyfmt_event is used during copyfmt().

5.223.4 Constructor & Destructor Documentation

basic_ios() [1/2]

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::basic_ios (
            basic_streambuf< _CharT, _Traits > * __sb)  [inline], [explicit]
```

Constructor performs initialization.
The parameter is passed by derived streams.
Referenced by copyfmt().

~basic_ios()

```
template<typename _CharT, typename _Traits>
virtual std::basic_ios< _CharT, _Traits >::~basic_ios ()  [inline], [virtual]
```

Empty.
The destructor does nothing. More specifically, it does not destroy the streambuf held by rdbuf().

basic_ios() [2/2]

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::basic_ios ()  [inline], [protected]
```

Empty.
The default constructor does nothing and is not normally accessible to users.

5.223.5 Member Function Documentation

_M_getloc()

```
const locale & std::ios_base::_M_getloc () const  [inline], [inherited]
```

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.
Referenced by std::money_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter
std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIte
std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::num_put< _CharT, _OutIter
std::time_put< _CharT, _OutIter >::do_put(), std::time_get< _CharT, _InIter >::get(), and std::time_put< _CharT, _OutIter >::put().

bad()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad () const  [inline], [nodiscard]
```

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry().

clear()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::clear (
            iostate __state = goodbit)
```

[Re]sets the error state.

Parameters

__state	The new state flag(s) to set.

See std::ios_base::iostate for the possible bit values. Most users will not need to pass an argument.
References std::ios_base::badbit, exceptions(), rdbuf(), and rdstate().
Referenced by std::basic_ios< char_type, traits_type >::exceptions(), std::__detail::operator>>(), std::basic_istream< _CharT, _Traits >
rdbuf(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char_type, traits
and std::basic_istream< _CharT, _Traits >::unget().

copyfmt()

```
template<typename _CharT, typename _Traits>
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
            const basic_ios< _CharT, _Traits > & __rhs)
```

Copies fields of __rhs into this.

Parameters

__rhs	The source values for the copies.

Returns

Reference to this object.

All fields of __rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase_event. After copying, each (new) callback is invoked with copyfmt_event. The final step is to copy exceptions().
References basic_ios(), std::__addressof(), exceptions(), fill(), std::ios_base::flags(), std::ios_base::getloc(), std::ios_base::precision(), tie(), std::tie(), and std::ios_base::width().

eof()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof () const  [inline], [nodiscard]
```

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

exceptions() [1/2]

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions () const  [inline], [nodiscard]
```

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.
Referenced by clear(), and copyfmt().

exceptions() [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
            iostate __except)  [inline]
```

Throwing exceptions on errors.

Parameters

__except	The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type std::ios_base::failure is thrown.
If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr « "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr « "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const  [inline], [nodiscard]
```

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.
Referenced by std::basic_ios< char_type, traits_type >::operator bool(), std::basic_ios< char_type, traits_type >::operator!(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >:: std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >:: and std::regex_traits< _Ch_type >::value().

fill() [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const  [inline], [nodiscard]
```

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.
Referenced by copyfmt().

fill() [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
            char_type __ch)  [inline]
```

Sets a new *empty* character.

Parameters

__ch	The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

flags() [1/2]

```
fmtflags std::ios_base::flags () const  [inline], [nodiscard], [inherited]
```

Access to format flags.

Returns

The format control flags for both input and output.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter > std::num_put< _CharT, _OutIter >::do_put(), std::num_put< _CharT, _OutIter >::do_put(), std::chrono::operator<<(), std::operator<<(), std::__detail::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), and std::operator>>().

flags() [2/2]

```
fmtflags std::ios_base::flags (
            fmtflags __fmtfl)  [inline], [inherited]
```

Setting new format flags all at once.

Parameters

__fmtfl	The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *__fmtfl*.
References fmtflags.

getloc()

```
locale std::ios_base::getloc () const  [inline], [nodiscard], [inherited]
```

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::money_put< _CharT, _OutIter >::do_put(), std::basic_ios< _CharT, _Tr
std::chrono::operator<<(), std::operator>>(), std::operator>>(), and std::ws().

good()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good () const  [inline], [nodiscard]
```

Fast error checking.

Returns

True if no error flags are set.

A wrapper around rdstate.
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::__detail::operator>>().

imbue()

```
template<typename _CharT, typename _Traits>
locale std::basic_ios< _CharT, _Traits >::imbue (
            const locale & __loc)
```

Moves to a new locale.

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.
Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html
References std::ios_base::getloc(), std::ios_base::imbue(), and rdbuf().
Referenced by std::chrono::operator<<().

init()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::init (
            basic_streambuf< _CharT, _Traits > * __sb)  [protected]
```

All setup is performed here.
This is called from the public constructor. It is not virtual and cannot be redefined.
Referenced by std::basic_ios< char_type, traits_type >::basic_ios(), and std::basic_ostream< char_type, traits_type >::basic_ostream().

iword()

```
long & std::ios_base::iword (
            int __ix)  [inline], [inherited]
```

Access to integer array.

Parameters

_↩ _ix	Index into the array.

Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.
References iword().
Referenced by iword().

narrow()

```
template<typename _CharT, typename _Traits>
char std::basic_ios< _CharT, _Traits >::narrow (
            char_type __c,
            char __dfault) const  [inline]
```

Squeezes characters.

Parameters

__c	The character to narrow.
__dfault	The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.
Returns the result of
`std::use_facet<ctype<char_type> >(getloc()).narrow(c,dfault)`
Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.↩html

operator bool()

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool () const  [inline], [explicit], [nodiscard]
```

The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

operator"!()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator!  () const  [inline], [nodiscard]
```

The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

precision() [1/2]

```
streamsize std::ios_base::precision () const  [inline], [nodiscard], [inherited]
```

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::chrono::operator<<().

precision() [2/2]

```
streamsize std::ios_base::precision (
            streamsize __prec)  [inline], [inherited]
```

Changing flags.

Parameters

__prec	The new precision value.

Returns

The previous value of precision().

pword()

```
void *& std::ios_base::pword (
            int __ix)  [inline], [inherited]
```

Access to void pointer array.

Parameters

_↩ _ix	Index into the array.

Returns

A reference to a void* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

rdbuf() [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf () const  [inline],
[nodiscard]
```

Accessing the underlying buffer.

Returns

The current stream buffer.

This does not change the state of the stream.
Referenced by clear(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getlin std::getline(), std::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::ignore(), imbue(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::tr2::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >: std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >: std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::t std::basic_ostream< _CharT, _Traits >::tellp(), and std::ws().

rdbuf() [2/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
            basic_streambuf< _CharT, _Traits > * __sb)
```

Changing the underlying buffer.

Parameters

__sb	The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.
Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream     foo;         // or some other derived type
std::streambuf*  p = .....;

foo.ios::rdbuf(p);            // ios == basic_ios<char>
```

References clear().

rdstate()

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::rdstate () const  [inline], [nodiscard]
```

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See std::ios_base::iostate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().
Referenced by std::basic_ios< char_type, traits_type >::bad(), clear(), std::basic_ios< char_type, traits_type >::eof(), std::basic_ios< char_type, traits_type >::fail(), std::basic_ios< char_type, traits_type >::good(), std::basic_istream< _CharT, _Traits >::
std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char_type, traits_type >
and std::basic_istream< _CharT, _Traits >::unget().

register_callback()

```
void std::ios_base::register_callback (
            event_callback __fn,
            int __index)  [inherited]
```

Add the callback __fn with parameter __index.

Parameters

__fn	The function to add.
__index	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.
References fmtflags.

setf() [1/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl)  [inline], [inherited]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.
References fmtflags.
Referenced by std::__detail::operator>>().

setf() [2/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl,
            fmtflags __mask)  [inline], [inherited]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.
__mask	The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.
References fmtflags.

setstate()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
             iostate __state)  [inline]
```

Sets additional flags in the error state.

Parameters

__state	The additional state flag(s) to set.

See std::ios_base::iostate for the possible bit values.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::getline(), std::getline(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _Cha std::operator>>(), std::operator>>(), std::operator>>(), std::tr2::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >: std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits > std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), and std::ws().

sync_with_stdio()

```
static bool std::ios_base::sync_with_stdio (
             bool __sync = true)  [static], [inherited]
```

Interaction with the standard C I/O objects.

Parameters

__sync	Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary

tie() [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie () const  [inline],
[nodiscard]
```

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and copyfmt().

tie() [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie (
            basic_ostream< _CharT, _Traits > * __tiestr)  [inline]
```

Ties this stream to an output stream.

Parameters

__tiestr	The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

unsetf()

```
void std::ios_base::unsetf (
            fmtflags __mask)  [inline], [inherited]
```

Clearing format flags.

Parameters

__mask	The flags to unset.

This function clears *__mask* in the format flags.
References fmtflags.

widen()

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
            char __c) const  [inline]
```

Widens characters.

Parameters

_c	The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.
Returns the result of
`std::use_facet<ctype<char_type> >(getloc()).widen(c)`
Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html
Referenced by std::basic_ios< char_type, traits_type >::fill(), std::getline(), std::getline(), and std::tr2::operator>>().

width() [1/2]

```
streamsize std::ios_base::width () const  [inline], [nodiscard], [inherited]
```
Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_put< _CharT, _OutIter >::do_put(), std::operator>>(), and std::operator>>().

width() [2/2]

```
streamsize std::ios_base::width (
            streamsize __wide)  [inline], [inherited]
```
Changing flags.

Parameters

__wide	The new width value.

Returns

The previous value of width().

xalloc()

```
static int std::ios_base::xalloc () throw ( )   [static], [inherited]
```
Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the iword and pword functions. The expectation is that an application calls xalloc in order to obtain an index in the iword and pword arrays that can be used without fear of conflict.
The implementation maintains a static variable that is incremented and returned on each invocation. xalloc is guaranteed to return an index that is safe to use in the iword and pword arrays.

5.223.6 Member Data Documentation

adjustfield

```
const fmtflags std::ios_base::adjustfield  [static], [inherited]
```
A mask of left|right|internal. Useful for the 2-arg form of `setf`.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::internal(), std::left(), and std::right().

app

```
const openmode std::ios_base::app  [static], [inherited]
```
Seek to end before each write.
Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

ate

const openmode std::ios_base::ate [static], [inherited]
Open and seek to end immediately after opening.
Referenced by std::basic_filebuf< _CharT, _Traits >::open().

badbit

const iostate std::ios_base::badbit [static], [inherited]
Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< char_type, traits_type >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::basic_ios< _CharT, _Traits >::clear(), std::basic_ostream< _CharT, _Trai std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_ostream< _CharT, _Traits >::opera std::basic_ostream< char_type, traits_type >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >: std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< char_type, traits_ty std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_istream< char_type, traits_type and std::ws().

basefield

const fmtflags std::ios_base::basefield [static], [inherited]
A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.
Referenced by std::dec(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::hex(), and std::oct().

beg

const seekdir std::ios_base::beg [static], [inherited]
Request a seek relative to the beginning of the stream.
Referenced by std::basic_filebuf< _CharT, _Traits >::seekpos().

binary

const openmode std::ios_base::binary [static], [inherited]
Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.↩ filestreams.binary.
Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

boolalpha

const fmtflags std::ios_base::boolalpha [static], [inherited]
Insert/extract `bool` in alphabetic rather than numeric format.
Referenced by std::boolalpha(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::noboolalpha().

cur

const seekdir std::ios_base::cur [static], [inherited]
Request a seek relative to the current position within the sequence.
Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

dec

const fmtflags std::ios_base::dec [static], [inherited]
Converts integer input or generates integer output in decimal base.
Referenced by std::dec().

end

const seekdir std::ios_base::end [static], [inherited]
Request a seek relative to the current end of the sequence.
Referenced by std::basic_filebuf< _CharT, _Traits >::open(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff().

eofbit

const iostate std::ios_base::eofbit [static], [inherited]
Indicates that an input operation reached the end of an input sequence.
Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >:
std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(),
std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter
std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::ge
std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< char_type, traits_type >::g
std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::putback(),
std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< char_type, traits_type >::read(), std::basic_istream< _CharT, _Traits
std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< char_type, traits_ty
std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

failbit

const iostate std::ios_base::failbit [static], [inherited]
Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::num_get< _CharT, _InIter >::do_get(),
std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(),
std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::ge
std::basic_istream< char_type, traits_type >::get(), std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::oper
std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(),
std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::s
std::basic_ostream< _CharT, _Traits >::seekp(), and std::basic_ostream< _CharT, _Traits >::seekp().

fixed

const fmtflags std::ios_base::fixed [static], [inherited]
Generate floating-point output in fixed-point notation.
Referenced by std::fixed(), and std::hexfloat().

floatfield

const fmtflags std::ios_base::floatfield [static], [inherited]
A mask of scientific|fixed. Useful for the 2-arg form of `setf`.
Referenced by std::defaultfloat(), std::fixed(), std::hexfloat(), and std::scientific().

goodbit

const iostate std::ios_base::goodbit [static], [inherited]
Indicates all is well.

Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::d std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >: std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore() std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ostream< _CharT, _Traits > std::basic_ostream< char_type, traits_type >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >:: std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits std::basic_istream< char_type, traits_type >::seekg(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Trait and std::ws().

hex

```
const fmtflags std::ios_base::hex  [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.
Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::hex().

in

```
const openmode std::ios_base::in  [static], [inherited]
```

Open for input. Default for `ifstream` and fstream.
Referenced by std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_stringbuf< _CharT, _Traits, _ std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_filebuf< _CharT, _Traits >::underflow(), std::basic_stringbuf< _CharT, _Traits, _ and std::basic_filebuf< _CharT, _Traits >::xsgetn().

internal

```
const fmtflags std::ios_base::internal  [static], [inherited]
```

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.
Referenced by std::internal().

left

```
const fmtflags std::ios_base::left  [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)
Referenced by std::num_put< _CharT, _OutIter >::do_put(), and std::left().

oct

```
const fmtflags std::ios_base::oct  [static], [inherited]
```

Converts integer input or generates integer output in octal base.
Referenced by std::oct().

out

```
const openmode std::ios_base::out  [static], [inherited]
```

Open for output. Default for `ofstream` and fstream.
Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(),

std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_stringbuf< _CharT, _Traits, std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

right

`const fmtflags std::ios_base::right [static], [inherited]`
Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)
Referenced by std::right().

scientific

`const fmtflags std::ios_base::scientific [static], [inherited]`
Generates floating-point output in scientific notation.
Referenced by std::hexfloat(), and std::scientific().

showbase

`const fmtflags std::ios_base::showbase [static], [inherited]`
Generates a prefix indicating the numeric base of generated integer output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::noshowbase(), and std::showbase().

showpoint

`const fmtflags std::ios_base::showpoint [static], [inherited]`
Generates a decimal-point character unconditionally in generated floating-point output.
Referenced by std::noshowpoint(), and std::showpoint().

showpos

`const fmtflags std::ios_base::showpos [static], [inherited]`
Generates a + sign in non-negative generated numeric output.
Referenced by std::noshowpos(), and std::showpos().

skipws

`const fmtflags std::ios_base::skipws [static], [inherited]`
Skips leading white space before certain input operations.
Referenced by std::noskipws(), std::__detail::operator>>(), and std::skipws().

trunc

`const openmode std::ios_base::trunc [static], [inherited]`
Truncate an existing stream when opening. Default for `ofstream`.

unitbuf

`const fmtflags std::ios_base::unitbuf [static], [inherited]`
Flushes output after each output operation.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::nounitbuf(), and std::unitbuf().

uppercase

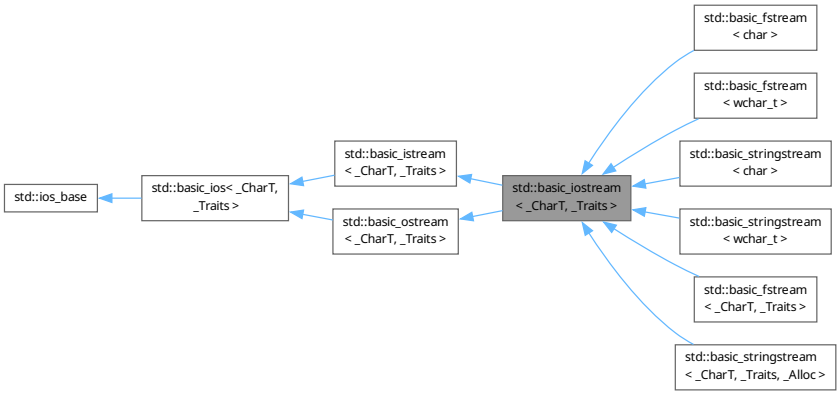
`const fmtflags std::ios_base::uppercase [static], [inherited]`
Replaces certain lowercase letters with their uppercase equivalents in generated output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::nouppercase(), and std::uppercase().
The documentation for this class was generated from the following files:

- iosfwd
- basic_ios.h
- basic_ios.tcc

5.224 std::basic_iostream< _CharT, _Traits > Class Template Reference

`#include <istream>`

Inheritance diagram for std::basic_iostream< _CharT, _Traits >:



Public Types

- typedef ctype< _CharT > **__ctype_type**
- typedef ctype< _CharT > **__ctype_type**
- typedef basic_ios< _CharT, _Traits > **__ios_type**
- typedef basic_ios< _CharT, _Traits > **__ios_type**
- typedef basic_istream< _CharT, _Traits > **__istream_type**
- typedef num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > **__num_get_type**
- typedef num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > **__num_put_type**
- typedef basic_ostream< _CharT, _Traits > **__ostream_type**
- typedef basic_streambuf< _CharT, _Traits > **__streambuf_type**
- typedef basic_streambuf< _CharT, _Traits > **__streambuf_type**
- typedef _CharT **char_type**
- enum event { **erase_event** , **imbue_event** , **copyfmt_event** }
- typedef void(* event_callback) (event __e, ios_base &__b, int __i)
- typedef _Traits::int_type **int_type**
- typedef _Ios_Iostate iostate
- typedef _Traits::off_type **off_type**
- typedef _Ios_Openmode openmode
- typedef _Traits::pos_type **pos_type**
- typedef _Ios_Seekdir seekdir
- typedef _Traits **traits_type**

Public Member Functions

- basic_iostream (basic_streambuf< _CharT, _Traits > *__sb)
- virtual ~basic_iostream ()
- template<typename _ValueT>
  basic_istream< _CharT, _Traits > & **_M_extract** (_ValueT &__v)
- const locale & _M_getloc () const
- template<typename _ValueT>
  basic_ostream< _CharT, _Traits > & **_M_insert** (_ValueT __v)
- void **_M_setstate** (iostate __state)
- bool bad () const
- void clear (iostate __state=goodbit)
- basic_ios & copyfmt (const basic_ios &__rhs)
- bool eof () const
- iostate exceptions () const
- void exceptions (iostate __except)
- bool fail () const
- char_type fill () const
- char_type fill (char_type __ch)
- fmtflags flags () const
- fmtflags flags (fmtflags __fmtfl)
- __ostream_type & flush ()
- streamsize gcount () const
- basic_istream< char > & getline (char_type *__s, streamsize __n, char_type __delim)
- basic_istream< wchar_t > & **getline** (char_type *__s, streamsize __n, char_type __delim)
- locale getloc () const
- bool good () const
- basic_istream< char > & **ignore** (streamsize __n)
- basic_istream< wchar_t > & **ignore** (streamsize __n)
- basic_istream< char > & **ignore** (streamsize __n, int_type __delim)
- basic_istream< wchar_t > & **ignore** (streamsize __n, int_type __delim)
- locale imbue (const locale &__loc)
- long & iword (int __ix)
- char narrow (char_type __c, char __dfault) const
- __ostream_type & operator<< (__streambuf_type *__sb)
- __ostream_type & operator<< (const void *__p)
- __ostream_type & **operator**<< (nullptr_t)
- __istream_type & operator>> (__streambuf_type *__sb)
- __istream_type & operator>> (void *&__p)
- streamsize precision () const
- streamsize precision (streamsize __prec)
- void *& pword (int __ix)
- basic_streambuf< _CharT, _Traits > * rdbuf () const
- basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > *__sb)
- iostate rdstate () const
- void register_callback (event_callback __fn, int __index)
- __ostream_type & seekp (off_type, ios_base::seekdir)
- __ostream_type & seekp (pos_type)
- fmtflags setf (fmtflags __fmtfl)
- fmtflags setf (fmtflags __fmtfl, fmtflags __mask)
- void setstate (iostate __state)

- pos_type tellp ()
- basic_ostream< _CharT, _Traits > * tie () const
- basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > *__tiestr)
- void unsetf (fmtflags __mask)
- char_type widen (char __c) const
- streamsize width () const
- streamsize width (streamsize __wide)

- __istream_type & operator>> (__istream_type &(*__pf)(__istream_type &))
- __istream_type & operator>> (__ios_type &(*__pf)(__ios_type &))
- __istream_type & operator>> (ios_base &(*__pf)(ios_base &))

Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type std::basic_istream::sentry with the second argument (noskipws) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, ios_base::badbit will be turned on in the stream's error state (without causing an ios_base::failure to be thrown) and the original exception will be rethrown if badbit is set in the exceptions mask.

- __istream_type & operator>> (bool &__n)
- __istream_type & operator>> (short &__n)
- __istream_type & operator>> (unsigned short &__n)
- __istream_type & operator>> (int &__n)
- __istream_type & operator>> (unsigned int &__n)
- __istream_type & operator>> (long &__n)
- __istream_type & operator>> (unsigned long &__n)
- __istream_type & operator>> (long long &__n)
- __istream_type & operator>> (unsigned long long &__n)

- __istream_type & operator>> (float &__f)
- __istream_type & operator>> (double &__f)
- __istream_type & operator>> (long double &__f)

Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type std::basic_istream::sentry with the second argument (noskipws) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by gcount().

If an exception is thrown during extraction, ios_base::badbit will be turned on in the stream's error state (without causing an ios_base::failure to be thrown) and the original exception will be rethrown if badbit is set in the exceptions mask.

- int_type get ()
- __istream_type & get (char_type &__c)
- __istream_type & get (char_type *__s, streamsize __n, char_type __delim)
- __istream_type & get (char_type *__s, streamsize __n)
- __istream_type & get (__streambuf_type &__sb, char_type __delim)
- __istream_type & get (__streambuf_type &__sb)
- __istream_type & getline (char_type *__s, streamsize __n, char_type __delim)
- __istream_type & getline (char_type *__s, streamsize __n)

- __istream_type & ignore (streamsize __n, int_type __delim)
- __istream_type & ignore (streamsize __n)
- __istream_type & ignore ()
- int_type peek ()
- __istream_type & read (char_type *__s, streamsize __n)
- streamsize readsome (char_type *__s, streamsize __n)
- __istream_type & putback (char_type __c)
- __istream_type & unget ()
- int sync ()
- pos_type tellg ()
- __istream_type & seekg (pos_type)
- __istream_type & seekg (off_type, ios_base::seekdir)

- __ostream_type & operator<< (__ostream_type &(*__pf)(__ostream_type &))
- __ostream_type & operator<< (__ios_type &(*__pf)(__ios_type &))
- __ostream_type & operator<< (ios_base &(*__pf)(ios_base &))

Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type std::basic_ostream::sentry. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, ios_base::badbit will be turned on in the stream's error state without causing an ios_base::failure to be thrown. The original exception will then be rethrown.

- __ostream_type & operator<< (long __n)
- __ostream_type & operator<< (unsigned long __n)
- __ostream_type & operator<< (bool __n)
- __ostream_type & operator<< (short __n)
- __ostream_type & operator<< (unsigned short __n)
- __ostream_type & operator<< (int __n)
- __ostream_type & operator<< (unsigned int __n)
- __ostream_type & operator<< (long long __n)
- __ostream_type & operator<< (unsigned long long __n)

- __ostream_type & operator<< (double __f)
- __ostream_type & operator<< (float __f)
- __ostream_type & operator<< (long double __f)

Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type std::basic_ostream::sentry. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, ios_base::badbit will be turned on in the stream's error state. If badbit is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- __ostream_type & put (char_type __c)
- __ostream_type & write (const char_type *__s, streamsize __n)

- operator bool () const
- bool operator! () const

Static Public Member Functions

- static bool sync_with_stdio (bool __sync=true)
- static int xalloc () throw ()

Public Attributes

- class __attribute((__abi_tag__("cxx11"))) failure typedef _Ios_Fmtflags fmtflags

Static Public Attributes

- static const openmode **__noreplace**
- static const fmtflags adjustfield
- static const openmode app
- static const openmode ate
- static const iostate badbit
- static const fmtflags basefield
- static const seekdir beg
- static const openmode binary
- static const fmtflags boolalpha
- static const seekdir cur
- static const fmtflags dec
- static const seekdir end
- static const iostate eofbit
- static const iostate failbit
- static const fmtflags fixed
- static const fmtflags floatfield
- static const iostate goodbit
- static const fmtflags hex
- static const openmode in
- static const fmtflags internal
- static const fmtflags left
- static const fmtflags oct
- static const openmode out
- static const fmtflags right
- static const fmtflags scientific
- static const fmtflags showbase
- static const fmtflags showpoint
- static const fmtflags showpos
- static const fmtflags skipws
- static const openmode trunc
- static const fmtflags unitbuf
- static const fmtflags uppercase

Protected Types

- enum { **_S_local_word_size** }

Protected Member Functions

- **basic_iostream** (basic_iostream &&__rhs)
- **basic_iostream** (const basic_iostream &)=delete
- void **_M_cache_locale** (const locale &__loc)
- void **_M_call_callbacks** (event __ev) throw ()
- void **_M_dispose_callbacks** (void) throw ()
- template<typename _ValueT>
  __istream_type & **_M_extract** (_ValueT &__v)
- _Words & **_M_grow_words** (int __index, bool __iword)
- void **_M_init** () throw ()
- template<typename _ValueT>
  __ostream_type & **_M_insert** (_ValueT __v)
- void **_M_move** (ios_base &) noexcept
- void **_M_swap** (ios_base &__rhs) noexcept
- void init (basic_streambuf< _CharT, _Traits > *__sb)
- void **move** (basic_ios &&__rhs)
- void **move** (basic_ios &__rhs)
- basic_iostream & **operator=** (basic_iostream &&__rhs)
- basic_iostream & **operator=** (const basic_iostream &)=delete
- void **set_rdbuf** (basic_streambuf< _CharT, _Traits > *__sb)
- void **swap** (basic_ios &__rhs) noexcept
- void **swap** (basic_iostream &__rhs)
- void **swap** (basic_istream &__rhs)
- void **swap** (basic_ostream &__rhs)

Protected Attributes

- _Callback_list * **_M_callbacks**
- const __ctype_type * **_M_ctype**
- iostate **_M_exception**
- char_type **_M_fill**
- bool **_M_fill_init**
- fmtflags **_M_flags**
- streamsize _M_gcount
- locale **_M_ios_locale**
- _Words **_M_local_word** [_S_local_word_size]
- const __num_get_type * **_M_num_get**
- const __num_put_type * **_M_num_put**
- streamsize **_M_precision**
- basic_streambuf< _CharT, _Traits > * **_M_streambuf**
- iostate **_M_streambuf_state**
- basic_ostream< _CharT, _Traits > * **_M_tie**
- streamsize **_M_width**
- _Words * **_M_word**
- int **_M_word_size**
- _Words **_M_word_zero**

5.224.1 Detailed Description

template<typename _CharT, typename _Traits>
class std::basic_iostream< _CharT, _Traits >

Template class basic_iostream.

Template Parameters

_CharT	Type of character stream.
_Traits	Traits for character type, defaults to char_traits<_CharT>.

This class multiply inherits from the input and output stream classes simply to provide a single interface.

5.224.2 Member Typedef Documentation

event_callback

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i)  [inherited]
```

The type of an event callback function.

Parameters

_e	One of the members of the event enum.
_b	Reference to the ios_base object.
_i	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several ios_base and basic_ios functions, specifically imbue(), copyfmt(), and ~ios().

iostate

```
typedef _Ios_Iostate std::ios_base::iostate  [inherited]
```

This is a bitmask type.
_Ios_Iostate is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit
- eofbit
- failbit
- goodbit

openmode

```
typedef _Ios_Openmode std::ios_base::openmode  [inherited]
```

This is a bitmask type.
_Ios_Openmode is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type openmode are:

- app
- ate
- binary
- in
- out
- trunc

seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir  [inherited]
```

This is an enumerated type.
_Ios_Seekdir is implementation-defined. Defined values of type seekdir are:

- beg
- cur, equivalent to SEEK_CUR in the C standard library.
- end, equivalent to SEEK_END in the C standard library.

5.224.3 Member Enumeration Documentation

event

```
enum std::ios_base::event  [inherited]
```

The set of events that may be passed to an event callback.
erase_event is used during ~ios() and copyfmt(). imbue_event is used during imbue(). copyfmt_event is used during copyfmt().

5.224.4 Constructor & Destructor Documentation

basic_iostream()

```
template<typename _CharT, typename _Traits>
std::basic_iostream< _CharT, _Traits >::basic_iostream (
            basic_streambuf< _CharT, _Traits > * __sb)  [inline], [explicit]
```

Constructor does nothing.
Both of the parent classes are initialized with the same streambuf pointer passed to this constructor.

~basic_iostream()

```
template<typename _CharT, typename _Traits>
virtual std::basic_iostream< _CharT, _Traits >::~basic_iostream ()  [inline], [virtual]
```

Destructor does nothing.

5.224.5 Member Function Documentation

_M_getloc()

```
const locale & std::ios_base::_M_getloc () const  [inline], [inherited]
```

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.
Referenced by std::money_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter
std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIte
std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::num_put< _CharT, _OutIter
std::time_put< _CharT, _OutIter >::do_put(), std::time_get< _CharT, _InIter >::get(), and std::time_put< _CharT, _OutIter >::put().

bad()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry().

clear()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::clear (
             iostate __state = goodbit)  [inherited]
```

[Re]sets the error state.

Parameters

__state	The new state flag(s) to set.

See std::ios_base::iostate for the possible bit values. Most users will not need to pass an argument.
References std::ios_base::badbit, exceptions(), rdbuf(), and rdstate().
Referenced by std::basic_ios< char_type, traits_type >::exceptions(), std::__detail::operator>>(), std::basic_istream< _CharT, _Traits >
rdbuf(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char_type, traits
and std::basic_istream< _CharT, _Traits >::unget().

copyfmt()

```
template<typename _CharT, typename _Traits>
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
             const basic_ios< _CharT, _Traits > & __rhs)  [inherited]
```

Copies fields of __rhs into this.

Parameters

__rhs	The source values for the copies.

Returns

Reference to this object.

All fields of __rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase_event. After copying, each (new) callback is invoked with copyfmt_event. The final step is to copy exceptions().
References basic_ios(), std::__addressof(), exceptions(), fill(), std::ios_base::flags(), std::ios_base::getloc(), std::ios_base::precision(), tie(), std::tie(), and std::ios_base::width().

eof()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

exceptions() [1/2]

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions () const  [inline], [nodiscard], [inherited]
```

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.
Referenced by clear(), and copyfmt().

exceptions() [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
            iostate __except)  [inline], [inherited]
```

Throwing exceptions on errors.

Parameters

__except	The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type std::ios_base::failure is thrown.
If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr « "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr « "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.
Referenced by std::basic_ios< char_type, traits_type >::operator bool(), std::basic_ios< char_type, traits_type >::operator!(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >:: std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >:: and std::regex_traits< _Ch_type >::value().

fill() [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const  [inline], [nodiscard], [inherited]
```

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.
Referenced by copyfmt().

fill() [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
            char_type __ch)  [inline], [inherited]
```

Sets a new *empty* character.

Parameters

__ch	The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

flags() [1/2]

```
fmtflags std::ios_base::flags () const  [inline], [nodiscard], [inherited]
```

Access to format flags.

Returns

The format control flags for both input and output.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >
std::num_put< _CharT, _OutIter >::do_put(), std::num_put< _CharT, _OutIter >::do_put(), std::chrono::operator<<(),
std::operator<<(), std::__detail::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(),
std::operator>>(), std::operator>>(), and std::operator>>().

flags() [2/2]

```
fmtflags std::ios_base::flags (
            fmtflags __fmtfl)  [inline], [inherited]
```

Setting new format flags all at once.

Parameters

__fmtfl	The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *__fmtfl*.
References fmtflags.

flush()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::flush ()  [inherited]
```

Synchronizing the stream buffer.

Returns

*this

If `rdbuf()` is a null pointer, changes nothing.
Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit.
References std::ios_base::badbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits

gcount()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::gcount () const  [inline], [inherited]
```

Character counting.

Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

get() [1/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::get (
            void )  [inherited]
```

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().
References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

get() [2/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::get (
            __streambuf_type & __sb)  [inline], [inherited]
```

Extraction into another streambuf.

Parameters

__sb	A streambuf in which to store data.

Returns

*this

Returns `get`(__sb,widen('\n')).

get() [3/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
            __streambuf_type & __sb,
            char_type __delim)  [inherited]
```

Extraction into another streambuf.

Parameters

__sb	A streambuf in which to store data.
__delim	A "stop" character.

Returns

*this

Characters are extracted and inserted into *__sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals *__delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.
References _M_gcount, and std::ios_base::goodbit.

get() [4/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
            char_type & __c)  [inherited]
```

Simple extraction.

Parameters

__c	The character in which to store data.

Returns

*this

Tries to extract a character and store it in *__c*. If none are available, sets failbit and returns traits::eof().

Note

This function is not overloaded on signed char and unsigned char.

References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

get() [5/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::get (
            char_type * __s,
            streamsize __n)  [inline], [inherited]
```

Simple multiple-character extraction.

Parameters

_↩ _s	Pointer to an array.
_↩ _n	Maximum number of characters to store in *s*.

Returns

*this

Returns `get(__s,__n,widen('\n'))`.

get() [6/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
            char_type * __s,
            streamsize __n,
            char_type __delim)  [inherited]
```

Simple multiple-character extraction.

Parameters

__s	Pointer to an array.
__n	Maximum number of characters to store in *__s*.
__delim	A "stop" character.

Returns

*this

Characters are extracted and stored into *__s* until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals *__delim*, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.
In any case, a null character is stored into the next location in the array.

Note

This function is not overloaded on signed char and unsigned char.

References _M_gcount, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_streambuf< _CharT, _Traits >:

getline() [1/3]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::getline (
            char_type * __s,
            streamsize __n)  [inline], [inherited]
```

String extraction.

Parameters

_↩ _s	A character array in which to store the data.
_↩ _n	Maximum number of characters to extract.

Returns

*this

Returns `getline(__s,__n,widen('\n'))`.

getline() [2/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::getline (
             char_type * __s,
             streamsize __n,
             char_type __delim)  [inherited]
```

String extraction.

Parameters

__s	A character array in which to store the data.
__n	Maximum number of characters to extract.
__delim	A "stop" character.

Returns

*this

Extracts and stores characters into *__s* until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the *__s* array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)
In any case, a null character is stored in the next location in the array.
References _M_gcount, std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_streambuf< _CharT, _Traits >::sbumpc(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _Cha

getline() [3/3]

```
basic_istream< char > & std::basic_istream< char >::getline (
             char_type * __s,
             streamsize __n,
             char_type __delim)  [inherited]
```

Explicit specialization declarations, defined in src/istream.cc.

getloc()

```
locale std::ios_base::getloc () const  [inline], [nodiscard], [inherited]
```
Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::money_put< _CharT, _OutIter >::do_put(), std::basic_ios< _CharT, _Tr
std::chrono::operator<<(), std::operator>>(), std::operator>>(), and std::ws().

good()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good () const  [inline], [nodiscard], [inherited]
```
Fast error checking.

Returns

True if no error flags are set.

A wrapper around rdstate.
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::__detail::operator>>().

ignore() [1/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
            void )  [inherited]
```
Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().
References _M_gcount, std::ios_base::goodbit, and std::basic_ios< _CharT, _Traits >::rdbuf().

ignore() [2/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
            streamsize __n)  [inherited]
```
Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().
References _M_gcount, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_streambuf< _CharT, _Traits >:

ignore() [3/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
            streamsize __n,
            int_type __delim)  [inherited]
```

Discarding characters.

Parameters

__n	Number of characters to discard.
__delim	A "stop" character.

Returns

*this

Extracts characters and throws them away until one of the following happens:

- if *__n* != `std::numeric_limits<int>::max()`, *__n* characters are extracted
- the input sequence reaches end-of-file
- the next character equals *__delim* (in this case, the character is extracted); note that this condition will never occur if *__delim* equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.
References _M_gcount, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_streambuf< _CharT, _Traits >:

imbue()

```
template<typename _CharT, typename _Traits>
locale std::basic_ios< _CharT, _Traits >::imbue (
            const locale & __loc)  [inherited]
```

Moves to a new locale.

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.
Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html
References std::ios_base::getloc(), std::ios_base::imbue(), and rdbuf().
Referenced by std::chrono::operator<<().

init()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::init (
            basic_streambuf< _CharT, _Traits > * __sb)  [protected], [inherited]
```

All setup is performed here.
This is called from the public constructor. It is not virtual and cannot be redefined.
Referenced by std::basic_ios< char_type, traits_type >::basic_ios(), and std::basic_ostream< char_type, traits_type >::basic_ostream().

iword()

```
long & std::ios_base::iword (
            int __ix)  [inline], [inherited]
```

Access to integer array.

Parameters

_↩ _*ix*	Index into the array.

Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.
References iword().
Referenced by iword().

narrow()

```
template<typename _CharT, typename _Traits>
char std::basic_ios< _CharT, _Traits >::narrow (
            char_type __c,
            char __dfault) const  [inline], [inherited]
```

Squeezes characters.

Parameters

__c	The character to narrow.
__dfault	The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.
Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c,dfault)
```

Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.↩html

operator bool()

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool () const  [inline], [explicit], [nodiscard],
[inherited]
```

The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

operator"!()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator!  () const  [inline], [nodiscard], [inherited]
```

The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

operator<<() [1/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            __ios_type &(* __pf )(__ios_type &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

operator<<() [2/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            __ostream_type &(* __pf )(__ostream_type &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

operator<<() [3/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
            __streambuf_type * __sb)  [inherited]
```

Extracting from another streambuf.

Parameters

__sb	A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.
If `__sb` is NULL, the stream will set failbit in its error state.
Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets failbit in the error state

If the function inserts no characters, failbit is set.

operator<<() [4/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            bool __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [5/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            const void * __p)  [inline], [inherited]
```

Pointer arithmetic inserters.

Parameters

_↩ _p	A variable of pointer type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [6/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            double __f)  [inline], [inherited]
```

Floating point arithmetic inserters.

Parameters

↩ _↩ ↩ _↩ f	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [7/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            float __f)  [inline], [inherited]
```

Floating point arithmetic inserters.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [8/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
            int __n)  [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _*n*	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [9/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            ios_base &(* __pf )(ios_base &))  [inline], [inherited]
```

Interface for manipulators.

Manipulators such as std::endl and std::hex use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

operator<<() [10/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            long __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _*n*	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [11/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            long double __f)  [inline], [inherited]
```

Floating point arithmetic inserters.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [12/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            long long __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ *_n*	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [13/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
            short __n)  [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ *_n*	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.
References basic_ostream(), std::ios_base::badbit, std::ostreambuf_iterator< _CharT, _Traits >::failed(), std::ios_base::goodbit, std::num_put< _CharT, _OutIter >::put(), std::basic_ios< _CharT, _Traits >::setstate(), and std::use_facet().

operator<<() [14/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned int __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [15/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned long __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [16/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned long long __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [17/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned short __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator>>() [1/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            __ios_type &(* __pf )(__ios_type &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as std::ws and std::dec use these functions in constructs like std::cin >> std::ws. For more information, see the iomanip header.

operator>>() [2/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            __istream_type &(* __pf )(__istream_type &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as std::ws and std::dec use these functions in constructs like std::cin >> std::ws. For more information, see the iomanip header.

operator>>() [3/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
            __streambuf_type * __sb)  [inherited]
```

Extracting into another streambuf.

Parameters

__sb	A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.
If __sb is NULL, the stream will set failbit in its error state.
Characters are extracted from this stream and inserted into the __sb streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or

- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.
References std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

operator>>() [4/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            bool & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [5/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            double & __f)  [inline], [inherited]
```

Floating point arithmetic extractors.

Parameters

↩ _↩ ↩ _↩ f	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [6/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            float & __f)  [inline], [inherited]
```

Floating point arithmetic extractors.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

operator>>() [7/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
            int & __n)  [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ *_n*	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.
References std::ios_base::badbit, std::ios_base::failbit, std::num_get< _CharT, _InIter >::get(), std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::setstate(), and std::use_facet().

operator>>() [8/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            ios_base &(* __pf )(ios_base &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the iomanip header.

operator>>() [9/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            long & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ *_n*	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [10/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            long double & __f)  [inline], [inherited]
```

Floating point arithmetic extractors.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [11/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            long long & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ *_n*	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [12/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
            short & __n)  [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ *_n*	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.
References std::ios_base::badbit, std::ios_base::failbit, std::num_get< _CharT, _InIter >::get(), std::ios_base::goodbit, and std::use_facet().

operator>>() [13/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned int & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [14/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned long & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [15/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned long long & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [16/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned short & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [17/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            void *& __p)  [inline], [inherited]
```

Basic arithmetic extractors.

Parameters

_↩ _p	A variable of pointer type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

peek()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::peek (
            void )  [inherited]
```

Looking ahead in the stream.

Returns

The next character, or eof().

If, after constructing the sentry object, good() is false, returns traits::eof(). Otherwise reads but does not extract the next input character.

precision() [1/2]

```
streamsize std::ios_base::precision () const  [inline], [nodiscard], [inherited]
```

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::chrono::operator<<().

precision() [2/2]

```
streamsize std::ios_base::precision (
            streamsize __prec)  [inline], [inherited]
```

Changing flags.

Parameters

__prec	The new precision value.

Returns

The previous value of precision().

put()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (
            char_type __c)  [inherited]
```

Simple insertion.

Parameters

_↩ _c	The character to insert.

Returns

*this

Tries to insert __c.

Note

This function is not overloaded on signed char and unsigned char.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits

putback()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::putback (
            char_type __c)  [inherited]
```

Unextracting a single character.

Parameters

_↩ _c	The character to push back into the input stream.

Returns

*this

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.
If `rdbuf()` is null or if `sputbackc()` fails, sets badbit in the error state.

Note

This function first clears eofbit. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References _M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::rdstate(), std::basic_ios< _CharT, _Traits >::setstate(), and std::basic_streambuf< _CharT, _Traits >::sputbackc().

pword()

```
void *& std::ios_base::pword (
            int __ix)  [inline], [inherited]
```

Access to void pointer array.

Parameters

_↩ _ix	Index into the array.

Returns

A reference to a void* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

rdbuf() [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf () const  [inline],
[nodiscard], [inherited]
```

Accessing the underlying buffer.

Returns

The current stream buffer.

This does not change the state of the stream.
Referenced by clear(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getlin
std::getline(), std::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::ignore(), imbue(), std::basic_istream< _CharT, _Traits >::operator>>(),

std::operator>>(), std::operator>>(), std::operator>>(), std::tr2::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >:: std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits > std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::t std::basic_ostream< _CharT, _Traits >::tellp(), and std::ws().

rdbuf() [2/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
            basic_streambuf< _CharT, _Traits > * __sb)  [inherited]
```

Changing the underlying buffer.

Parameters

__sb	The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.
Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream     foo;         // or some other derived type
std::streambuf*  p = .....;

foo.ios::rdbuf(p);            // ios == basic_ios<char>
```

References clear().

rdstate()

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::rdstate () const  [inline], [nodiscard], [inherited]
```

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See std::ios_base::iostate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().
Referenced by std::basic_ios< char_type, traits_type >::bad(), clear(), std::basic_ios< char_type, traits_type >::eof(), std::basic_ios< char_type, traits_type >::fail(), std::basic_ios< char_type, traits_type >::good(), std::basic_istream< _CharT, _Traits >:: std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char_type, traits_type > and std::basic_istream< _CharT, _Traits >::unget().

read()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::read (
            char_type * __s,
            streamsize __n)  [inherited]
```

Extraction without delimiters.

Parameters

_↩ _s	A character array.
_↩ _n	Maximum number of characters to store.

Returns

*this

If the stream state is `good()`, extracts characters and stores them into __*s* until one of the following happens:

- __*n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

Note

This function is not overloaded on signed char and unsigned char.

References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

readsome()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::readsome (
             char_type * __s,
             streamsize __n)  [inherited]
```

Extraction until the buffer is exhausted, but no more.

Parameters

_↩ _s	A character array.
_↩ _n	Maximum number of characters to store.

Returns

The number of characters extracted.

Extracts characters and stores them into __*s* depending on the number of characters remaining in the streambuf's buffer, `rdbuf()->in_avail()`, called `A` here:

- if `A == -1`, sets eofbit and extracts no characters
- if `A == 0`, extracts no characters
- if `A > 0`, extracts `min(A,n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.
References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::min(), std::basic_ios< _CharT, _Traits >::
and std::basic_ios< _CharT, _Traits >::setstate().

register_callback()

```
void std::ios_base::register_callback (
            event_callback __fn,
            int __index)  [inherited]
```

Add the callback __fn with parameter __index.

Parameters

__fn	The function to add.
__index	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered. References fmtflags.

seekg() [1/2]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
            off_type __off,
            ios_base::seekdir __dir)  [inherited]
```

Changing the current read position.

Parameters

__off	A file offset object.
__dir	The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(__off,__dir)`. If that function fails, sets failbit.

Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::basic_ios< _CharT, _Traits >::fail(
std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::in, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits
and std::basic_ios< _CharT, _Traits >::setstate().

seekg() [2/2]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
            pos_type __pos)  [inherited]
```

Changing the current read position.

Parameters

__pos	A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets failbit.

Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::in, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >, and std::basic_ios< _CharT, _Traits >::setstate().

seekp() [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
            off_type __off,
            ios_base::seekdir __dir)  [inherited]
```

Changing the current write position.

Parameters

__off	A file offset object.
__dir	The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off,dir)`. If that function fails, sets failbit.
References std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::out, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

seekp() [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
            pos_type __pos)  [inherited]
```

Changing the current write position.

Parameters

__pos	A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.
References std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::out, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

setf() [1/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl)  [inline], [inherited]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.
References fmtflags.
Referenced by std::__detail::operator>>().

setf() [2/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl,
            fmtflags __mask)  [inline], [inherited]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.
__mask	The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.
References fmtflags.

setstate()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
            iostate __state)  [inline], [inherited]
```

Sets additional flags in the error state.

Parameters

__state	The additional state flag(s) to set.

See std::ios_base::iostate for the possible bit values.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::getline(), std::getline(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _Cha std::operator>>(), std::operator>>(), std::operator>>(), std::tr2::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >: std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits > std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), and std::ws().

sync()

```
template<typename _CharT, typename _Traits>
int std::basic_istream< _CharT, _Traits >::sync (
            void )  [inherited]
```

Synchronizing the stream buffer.

Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.
Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit and returns -1.
Otherwise, returns 0.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_streambuf< _CharT, _Traits >::pubsync(), std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

sync_with_stdio()

```
static bool std::ios_base::sync_with_stdio (
            bool __sync = true)  [static], [inherited]
```

Interaction with the standard C I/O objects.

Parameters

__sync	Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary

tellg()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::pos_type std::basic_istream< _CharT, _Traits >::tellg (
            void )  [inherited]
```

Getting the current read position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type`(-1) to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0,cur,in)`.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with putback, unget and seekg, eofbit is not cleared first.

References std::ios_base::badbit, std::ios_base::cur, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::in, and std::basic_ios< _CharT, _Traits >::rdbuf().

tellp()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits >::pos_type std::basic_ostream< _CharT, _Traits >::tellp ()  [inherited]
```

Getting the current write position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type`(-1) to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0,cur,out)`.
References std::ios_base::cur, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::out, and std::basic_ios< _CharT, _Traits >::rdbuf()

tie() [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie () const  [inline],
[nodiscard], [inherited]
```

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and copyfmt().

tie() [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie (
            basic_ostream< _CharT, _Traits > * __tiestr)  [inline], [inherited]
```

Ties this stream to an output stream.

Parameters

__tiestr	The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

unget()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::unget (
            void )  [inherited]
```

Unextracting the previous character.

Returns

*this

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.
If `rdbuf()` is null or if `sungetc()` fails, sets badbit in the error state.

Note

This function first clears eofbit. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References _M_gcount, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::ios_base::goodbit, and std::basic_ios< _CharT, _Traits >::rdstate().
Referenced by std::__detail::operator>>().

unsetf()

```
void std::ios_base::unsetf (
             fmtflags __mask)  [inline], [inherited]
```

Clearing format flags.

Parameters

__mask	The flags to unset.

This function clears *__mask* in the format flags.
References fmtflags.

widen()

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
             char __c) const  [inline], [inherited]
```

Widens characters.

Parameters

_↩ _c	The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.
Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.↩ html
Referenced by std::basic_ios< char_type, traits_type >::fill(), std::getline(), std::getline(), and std::tr2::operator>>().

width() [1/2]

```
streamsize std::ios_base::width () const  [inline], [nodiscard], [inherited]
```

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_put< _CharT, _OutIter >::do_put(), std::operator>>(), and std::operator>>().

width() [2/2]

```
streamsize std::ios_base::width (
            streamsize __wide)  [inline], [inherited]
```

Changing flags.

Parameters

__wide	The new width value.

Returns

The previous value of width().

write()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::write (
            const char_type * __s,
            streamsize __n)  [inherited]
```

Character string insertion.

Parameters

_↩ _s	The array to insert.
_↩ _n	Maximum number of characters to insert.

Returns

*this

Characters are copied from __s and inserted into the stream until one of the following happens:

- __n characters are inserted
- inserting into the output sequence fails (in this case, badbit will be set in the stream's error state)

Note

This function is not overloaded on signed char and unsigned char.

xalloc()

```
static int std::ios_base::xalloc () throw ( )   [static], [inherited]
```

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the iword and pword functions. The expectation is that an application calls xalloc in order to obtain an index in the iword and pword arrays that can be used without fear of conflict.
The implementation maintains a static variable that is incremented and returned on each invocation. xalloc is guaranteed to return an index that is safe to use in the iword and pword arrays.

5.224.6 Member Data Documentation

_M_gcount

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::_M_gcount  [protected], [inherited]
```

The number of characters extracted in the previous unformatted function; see gcount().
Referenced by get(), get(), get(), get(), getline(), ignore(), ignore(), ignore(), putback(), read(), readsome(), and unget().

adjustfield

```
const fmtflags std::ios_base::adjustfield  [static], [inherited]
```

A mask of left|right|internal. Useful for the 2-arg form of `setf`.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::internal(), std::left(), and std::right().

app

```
const openmode std::ios_base::app  [static], [inherited]
```

Seek to end before each write.
Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

ate

```
const openmode std::ios_base::ate  [static], [inherited]
```

Open and seek to end immediately after opening.
Referenced by std::basic_filebuf< _CharT, _Traits >::open().

badbit

```
const iostate std::ios_base::badbit  [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< char_type, traits_type >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::basic_ios< _CharT, _Traits >::clear(), std::basic_ostream< _CharT, _Trai std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_ostream< _CharT, _Traits >::opera std::basic_ostream< char_type, traits_type >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >: std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< char_type, traits_ty std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_istream< char_type, traits_type and std::ws().

basefield

```
const fmtflags std::ios_base::basefield  [static], [inherited]
```

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.
Referenced by std::dec(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::hex(), and std::oct().

beg

```
const seekdir std::ios_base::beg  [static], [inherited]
```

Request a seek relative to the beginning of the stream.
Referenced by std::basic_filebuf< _CharT, _Traits >::seekpos().

binary

```
const openmode std::ios_base::binary  [static], [inherited]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary.

Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

boolalpha

```
const fmtflags std::ios_base::boolalpha  [static], [inherited]
```

Insert/extract `bool` in alphabetic rather than numeric format.

Referenced by std::boolalpha(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::noboolalpha().

cur

```
const seekdir std::ios_base::cur  [static], [inherited]
```

Request a seek relative to the current position within the sequence.

Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

dec

```
const fmtflags std::ios_base::dec  [static], [inherited]
```

Converts integer input or generates integer output in decimal base.

Referenced by std::dec().

end

```
const seekdir std::ios_base::end  [static], [inherited]
```

Request a seek relative to the current end of the sequence.

Referenced by std::basic_filebuf< _CharT, _Traits >::open(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff().

eofbit

```
const iostate std::ios_base::eofbit  [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >:
std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(),
std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter
std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::ge
std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< char_type, traits_type >::g
std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::putback(),
std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< char_type, traits_type >::read(), std::basic_istream< _CharT, _Traits
std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< char_type, traits_ty
std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

failbit

```
const iostate std::ios_base::failbit  [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(),

std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::ge
std::basic_istream< char_type, traits_type >::get(), std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::oper
std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(),
std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::s
std::basic_ostream< _CharT, _Traits >::seekp(), and std::basic_ostream< _CharT, _Traits >::seekp().

fixed

```
const fmtflags std::ios_base::fixed  [static], [inherited]
```

Generate floating-point output in fixed-point notation.
Referenced by std::fixed(), and std::hexfloat().

floatfield

```
const fmtflags std::ios_base::floatfield  [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.
Referenced by std::defaultfloat(), std::fixed(), std::hexfloat(), and std::scientific().

goodbit

```
const iostate std::ios_base::goodbit  [static], [inherited]
```

Indicates all is well.
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::num_get< _CharT, _InIter >::do_get(),
std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::d
std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >:
std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(),
std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore()
std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ostream< _CharT, _Traits >
std::basic_ostream< char_type, traits_type >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(),
std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(),
std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::
std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits
std::basic_istream< char_type, traits_type >::seekg(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Trait
and std::ws().

hex

```
const fmtflags std::ios_base::hex  [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.
Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and
std::hex().

in

```
const openmode std::ios_base::in  [static], [inherited]
```

Open for input. Default for `ifstream` and fstream.
Referenced by std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(),
std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_stringbuf< _CharT, _Traits, _
std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_filebuf< _CharT, _Traits >::showmanyc(),
std::basic_istream< _CharT, _Traits >::tellg(), std::basic_filebuf< _CharT, _Traits >::underflow(), std::basic_stringbuf< _CharT, _Traits, _
and std::basic_filebuf< _CharT, _Traits >::xsgetn().

internal

```
const fmtflags std::ios_base::internal  [static], [inherited]
```

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.
Referenced by std::internal().

left

`const fmtflags std::ios_base::left [static], [inherited]`
Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)
Referenced by std::num_put< _CharT, _OutIter >::do_put(), and std::left().

oct

`const fmtflags std::ios_base::oct [static], [inherited]`
Converts integer input or generates integer output in octal base.
Referenced by std::oct().

out

`const openmode std::ios_base::out [static], [inherited]`
Open for output. Default for `ofstream` and fstream.
Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_stringbuf< _CharT, _Traits, std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

right

`const fmtflags std::ios_base::right [static], [inherited]`
Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)
Referenced by std::right().

scientific

`const fmtflags std::ios_base::scientific [static], [inherited]`
Generates floating-point output in scientific notation.
Referenced by std::hexfloat(), and std::scientific().

showbase

`const fmtflags std::ios_base::showbase [static], [inherited]`
Generates a prefix indicating the numeric base of generated integer output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::noshowbase(), and std::showbase().

showpoint

`const fmtflags std::ios_base::showpoint [static], [inherited]`
Generates a decimal-point character unconditionally in generated floating-point output.
Referenced by std::noshowpoint(), and std::showpoint().

showpos

`const fmtflags std::ios_base::showpos [static], [inherited]`
Generates a + sign in non-negative generated numeric output.
Referenced by std::noshowpos(), and std::showpos().

skipws

```
const fmtflags std::ios_base::skipws  [static], [inherited]
```

Skips leading white space before certain input operations.
Referenced by std::noskipws(), std::__detail::operator>>(), and std::skipws().

trunc

```
const openmode std::ios_base::trunc  [static], [inherited]
```

Truncate an existing stream when opening. Default for `ofstream`.

unitbuf

```
const fmtflags std::ios_base::unitbuf  [static], [inherited]
```

Flushes output after each output operation.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::nounitbuf(), and std::unitbuf().

uppercase

```
const fmtflags std::ios_base::uppercase  [static], [inherited]
```

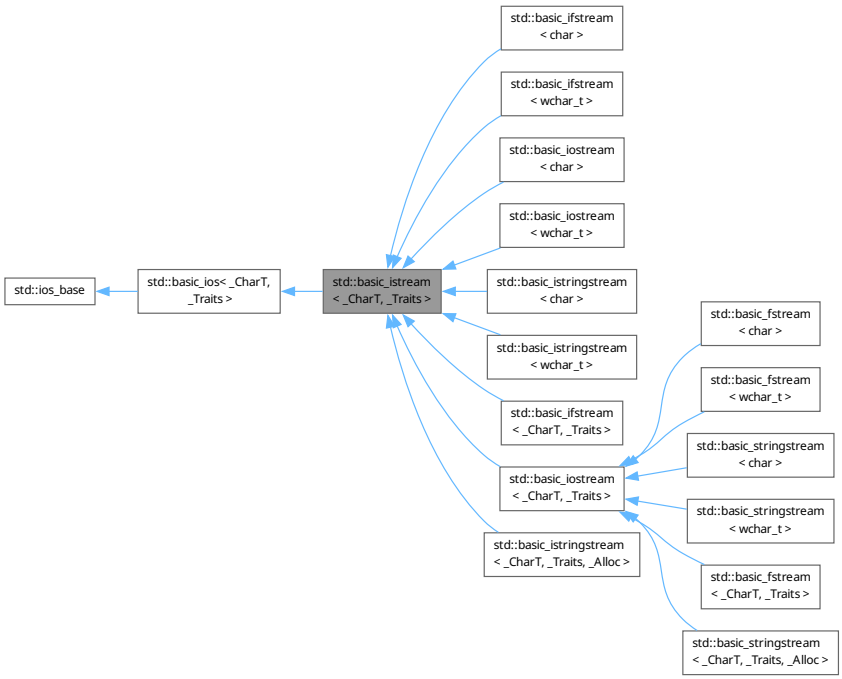
Replaces certain lowercase letters with their uppercase equivalents in generated output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::nouppercase(), and std::uppercase().
The documentation for this class was generated from the following files:

- iosfwd
- istream

5.225 std::basic_istream< _CharT, _Traits > Class Template Reference

```
#include <istream>
```

Inheritance diagram for std::basic_istream< _CharT, _Traits >:



Classes

- class sentry

Public Types

- typedef ctype< _CharT > **__ctype_type**
- typedef basic_ios< _CharT, _Traits > **__ios_type**
- typedef basic_istream< _CharT, _Traits > **__istream_type**
- typedef num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > **__num_get_type**
- typedef basic_streambuf< _CharT, _Traits > **__streambuf_type**
- typedef _CharT **char_type**
- enum event { **erase_event** , **imbue_event** , **copyfmt_event** }
- typedef void(* event_callback) (event __e, ios_base &__b, int __i)
- typedef _Traits::int_type **int_type**
- typedef _Ios_Iostate iostate
- typedef _Traits::off_type **off_type**
- typedef _Ios_Openmode openmode
- typedef _Traits::pos_type **pos_type**
- typedef _Ios_Seekdir seekdir
- typedef _Traits **traits_type**

- typedef num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > __num_put_type

Public Member Functions

- basic_istream (__streambuf_type *__sb)
- virtual ~basic_istream ()
- template<typename _ValueT>
  basic_istream< _CharT, _Traits > & **_M_extract** (_ValueT &__v)
- const locale & _M_getloc () const
- void **_M_setstate** (iostate __state)
- bool bad () const
- void clear (iostate __state=goodbit)
- basic_ios & copyfmt (const basic_ios &__rhs)
- bool eof () const
- iostate exceptions () const
- void exceptions (iostate __except)
- bool fail () const
- char_type fill () const
- char_type fill (char_type __ch)
- fmtflags flags () const
- fmtflags flags (fmtflags __fmtfl)
- streamsize gcount () const
- basic_istream< char > & getline (char_type *__s, streamsize __n, char_type __delim)
- basic_istream< wchar_t > & **getline** (char_type *__s, streamsize __n, char_type __delim)
- locale getloc () const
- bool good () const
- basic_istream< char > & **ignore** (streamsize __n)
- basic_istream< wchar_t > & **ignore** (streamsize __n)
- basic_istream< char > & **ignore** (streamsize __n, int_type __delim)
- basic_istream< wchar_t > & **ignore** (streamsize __n, int_type __delim)
- locale imbue (const locale &__loc)
- long & iword (int __ix)
- char narrow (char_type __c, char __dfault) const
- __istream_type & operator>> (__streambuf_type *__sb)
- __istream_type & operator>> (void *&__p)
- streamsize precision () const
- streamsize precision (streamsize __prec)
- void *& pword (int __ix)
- basic_streambuf< _CharT, _Traits > * rdbuf () const
- basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > *__sb)
- iostate rdstate () const
- void register_callback (event_callback __fn, int __index)
- fmtflags setf (fmtflags __fmtfl)
- fmtflags setf (fmtflags __fmtfl, fmtflags __mask)
- void setstate (iostate __state)
- basic_ostream< _CharT, _Traits > * tie () const
- basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > *__tiestr)
- void unsetf (fmtflags __mask)
- char_type widen (char __c) const
- streamsize width () const
- streamsize width (streamsize __wide)

- __istream_type & operator>> (__istream_type &(*__pf)(__istream_type &))

- __istream_type & operator>> (__ios_type &(*__pf)(__ios_type &))
- __istream_type & operator>> (ios_base &(*__pf)(ios_base &))

Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type std::basic_istream::sentry with the second argument (noskipws) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, ios_base::badbit will be turned on in the stream's error state (without causing an ios_base::failure to be thrown) and the original exception will be rethrown if badbit is set in the exceptions mask.

- __istream_type & operator>> (bool &__n)
- __istream_type & operator>> (short &__n)
- __istream_type & operator>> (unsigned short &__n)
- __istream_type & operator>> (int &__n)
- __istream_type & operator>> (unsigned int &__n)
- __istream_type & operator>> (long &__n)
- __istream_type & operator>> (unsigned long &__n)
- __istream_type & operator>> (long long &__n)
- __istream_type & operator>> (unsigned long long &__n)

- __istream_type & operator>> (float &__f)
- __istream_type & operator>> (double &__f)
- __istream_type & operator>> (long double &__f)

Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type std::basic_istream::sentry with the second argument (noskipws) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by gcount().

If an exception is thrown during extraction, ios_base::badbit will be turned on in the stream's error state (without causing an ios_base::failure to be thrown) and the original exception will be rethrown if badbit is set in the exceptions mask.

- int_type get ()
- __istream_type & get (char_type &__c)
- __istream_type & get (char_type *__s, streamsize __n, char_type __delim)
- __istream_type & get (char_type *__s, streamsize __n)
- __istream_type & get (__streambuf_type &__sb, char_type __delim)
- __istream_type & get (__streambuf_type &__sb)
- __istream_type & getline (char_type *__s, streamsize __n, char_type __delim)
- __istream_type & getline (char_type *__s, streamsize __n)
- __istream_type & ignore (streamsize __n, int_type __delim)
- __istream_type & ignore (streamsize __n)
- __istream_type & ignore ()
- int_type peek ()
- __istream_type & read (char_type *__s, streamsize __n)
- streamsize readsome (char_type *__s, streamsize __n)
- __istream_type & putback (char_type __c)
- __istream_type & unget ()
- int sync ()
- pos_type tellg ()
- __istream_type & seekg (pos_type)
- __istream_type & seekg (off_type, ios_base::seekdir)

- operator bool () const
- bool operator! () const

Static Public Member Functions

- static bool sync_with_stdio (bool __sync=true)
- static int xalloc () throw ()

Public Attributes

- class __attribute((__abi_tag__("cxx11"))) failure typedef _Ios_Fmtflags fmtflags

Static Public Attributes

- static const openmode **__noreplace**
- static const fmtflags adjustfield
- static const openmode app
- static const openmode ate
- static const iostate badbit
- static const fmtflags basefield
- static const seekdir beg
- static const openmode binary
- static const fmtflags boolalpha
- static const seekdir cur
- static const fmtflags dec
- static const seekdir end
- static const iostate eofbit
- static const iostate failbit
- static const fmtflags fixed
- static const fmtflags floatfield
- static const iostate goodbit
- static const fmtflags hex
- static const openmode in
- static const fmtflags internal
- static const fmtflags left
- static const fmtflags oct
- static const openmode out
- static const fmtflags right
- static const fmtflags scientific
- static const fmtflags showbase
- static const fmtflags showpoint
- static const fmtflags showpos
- static const fmtflags skipws
- static const openmode trunc
- static const fmtflags unitbuf
- static const fmtflags uppercase

Protected Types

- enum { **_S_local_word_size** }

Protected Member Functions

- **basic_istream** (basic_istream &&__rhs)
- **basic_istream** (const basic_istream &)=delete
- void **_M_cache_locale** (const locale &__loc)
- void **_M_call_callbacks** (event __ev) throw ()
- void **_M_dispose_callbacks** (void) throw ()
- template<typename _ValueT>
  __istream_type & **_M_extract** (_ValueT &__v)
- _Words & **_M_grow_words** (int __index, bool __iword)
- void **_M_init** () throw ()
- void **_M_move** (ios_base &) noexcept
- void **_M_swap** (ios_base &__rhs) noexcept
- void init (basic_streambuf< _CharT, _Traits > *__sb)
- void **move** (basic_ios &&__rhs)
- void **move** (basic_ios &__rhs)
- basic_istream & **operator=** (basic_istream &&__rhs)
- basic_istream & **operator=** (const basic_istream &)=delete
- void **set_rdbuf** (basic_streambuf< _CharT, _Traits > *__sb)
- void **swap** (basic_ios &__rhs) noexcept
- void **swap** (basic_istream &__rhs)

Protected Attributes

- _Callback_list * **_M_callbacks**
- const __ctype_type * **_M_ctype**
- iostate **_M_exception**
- char_type **_M_fill**
- bool **_M_fill_init**
- fmtflags **_M_flags**
- streamsize _M_gcount
- locale **_M_ios_locale**
- _Words **_M_local_word** [_S_local_word_size]
- const __num_get_type * **_M_num_get**
- const __num_put_type * **_M_num_put**
- streamsize **_M_precision**
- basic_streambuf< _CharT, _Traits > * **_M_streambuf**
- iostate **_M_streambuf_state**
- basic_ostream< _CharT, _Traits > * **_M_tie**
- streamsize **_M_width**
- _Words * **_M_word**
- int **_M_word_size**
- _Words **_M_word_zero**

Friends

- class **sentry**

5.225.1 Detailed Description

template<typename _CharT, typename _Traits>
class std::basic_istream< _CharT, _Traits >

Template class basic_istream.

Template Parameters

_CharT	Type of character stream.
_Traits	Traits for character type, defaults to char_traits<_CharT>.

This is the base class for all input streams. It provides text formatting of all builtin types, and communicates with any class derived from basic_streambuf to do the actual input.

5.225.2 Member Typedef Documentation

__num_put_type

```
template<typename _CharT, typename _Traits>
typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios< _CharT, _Traits
>::__num_put_type  [inherited]
```

These are non-standard types.

event_callback

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i)  [inherited]
```

The type of an event callback function.

Parameters

_e	One of the members of the event enum.
_b	Reference to the ios_base object.
_i	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several ios_base and basic_ios functions, specifically imbue(), copyfmt(), and ~ios().

iostate

```
typedef _Ios_Iostate std::ios_base::iostate  [inherited]
```

This is a bitmask type.
_Ios_Iostate is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit
- eofbit
- failbit
- goodbit

openmode

```
typedef _Ios_Openmode std::ios_base::openmode  [inherited]
```

This is a bitmask type.
_Ios_Openmode is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type openmode are:

- app
- ate
- binary
- in
- out
- trunc

seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir  [inherited]
```

This is an enumerated type.
_Ios_Seekdir is implementation-defined. Defined values of type seekdir are:

- beg
- cur, equivalent to SEEK_CUR in the C standard library.
- end, equivalent to SEEK_END in the C standard library.

5.225.3 Member Enumeration Documentation

event

```
enum std::ios_base::event  [inherited]
```

The set of events that may be passed to an event callback.
erase_event is used during ~ios() and copyfmt(). imbue_event is used during imbue(). copyfmt_event is used during copyfmt().

5.225.4 Constructor & Destructor Documentation

basic_istream()

```
template<typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits >::basic_istream (
            __streambuf_type * __sb)  [inline], [explicit]
```

Base constructor.
This ctor is almost never called by the user directly, rather from derived classes' initialization lists, which pass a pointer to their own stream buffer.

~basic_istream()

```
template<typename _CharT, typename _Traits>
virtual std::basic_istream< _CharT, _Traits >::~basic_istream ()  [inline], [virtual]
```

Base destructor.
This does very little apart from providing a virtual base dtor.

5.225.5 Member Function Documentation

_M_getloc()

```
const locale & std::ios_base::_M_getloc () const  [inline], [inherited]
```

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.
Referenced by std::money_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter
std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIte
std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::num_put< _CharT, _OutIter
std::time_put< _CharT, _OutIter >::do_put(), std::time_get< _CharT, _InIter >::get(), and std::time_put< _CharT, _OutIter >::put().

bad()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry().

clear()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::clear (
             iostate __state = goodbit)  [inherited]
```

[Re]sets the error state.

Parameters

__state	The new state flag(s) to set.

See std::ios_base::iostate for the possible bit values. Most users will not need to pass an argument.
References std::ios_base::badbit, exceptions(), rdbuf(), and rdstate().
Referenced by std::basic_ios< char_type, traits_type >::exceptions(), std::__detail::operator>>(), std::basic_istream< _CharT, _Traits >
rdbuf(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char_type, traits
and std::basic_istream< _CharT, _Traits >::unget().

copyfmt()

```
template<typename _CharT, typename _Traits>
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
             const basic_ios< _CharT, _Traits > & __rhs)  [inherited]
```

Copies fields of __rhs into this.

Parameters

__rhs	The source values for the copies.

Returns

Reference to this object.

All fields of __rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase_event. After copying, each (new) callback is invoked with copyfmt_event. The final step is to copy exceptions().
References basic_ios(), std::__addressof(), exceptions(), fill(), std::ios_base::flags(), std::ios_base::getloc(), std::ios_base::precision(), tie(), std::tie(), and std::ios_base::width().

eof()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

exceptions() [1/2]

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions () const  [inline], [nodiscard], [inherited]
```

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.
Referenced by clear(), and copyfmt().

exceptions() [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
            iostate __except)  [inline], [inherited]
```

Throwing exceptions on errors.

Parameters

__except	The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type std::ios_base::failure is thrown.
If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr « "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr « "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.
Referenced by std::basic_ios< char_type, traits_type >::operator bool(), std::basic_ios< char_type, traits_type >::operator!(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >:: std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >:: and std::regex_traits< _Ch_type >::value().

fill() [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const  [inline], [nodiscard], [inherited]
```

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.
Referenced by copyfmt().

fill() [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
            char_type __ch)  [inline], [inherited]
```

Sets a new *empty* character.

Parameters

__ch	The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

flags() [1/2]

```
fmtflags std::ios_base::flags () const  [inline], [nodiscard], [inherited]
```

Access to format flags.

Returns

The format control flags for both input and output.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter > std::num_put< _CharT, _OutIter >::do_put(), std::num_put< _CharT, _OutIter >::do_put(), std::chrono::operator<<(), std::operator<<(), std::__detail::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), and std::operator>>().

flags() [2/2]

```
fmtflags std::ios_base::flags (
            fmtflags __fmtfl)  [inline], [inherited]
```

Setting new format flags all at once.

Parameters

__fmtfl	The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *__fmtfl*.
References fmtflags.

gcount()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::gcount () const  [inline]
```

Character counting.

Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

get() [1/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::get (
            void )
```

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().
References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

get() [2/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::get (
            __streambuf_type & __sb)  [inline]
```

Extraction into another streambuf.

Parameters

__sb	A streambuf in which to store data.

Returns

*this

Returns get(__sb,widen('\n')).

get() [3/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
            __streambuf_type & __sb,
            char_type __delim)
```

Extraction into another streambuf.

Parameters

__sb	A streambuf in which to store data.
__delim	A "stop" character.

Returns

*this

Characters are extracted and inserted into *__sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals *__delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.
References _M_gcount, and std::ios_base::goodbit.

get() [4/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
            char_type & __c)
```

Simple extraction.

Parameters

__c	The character in which to store data.

Returns

*this

Tries to extract a character and store it in *__c*. If none are available, sets failbit and returns traits::eof().

Note

This function is not overloaded on signed char and unsigned char.

References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

get() [5/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::get (
            char_type * __s,
            streamsize __n)  [inline]
```

Simple multiple-character extraction.

Parameters

_↩ _*s*	Pointer to an array.
_↩ _*n*	Maximum number of characters to store in *s*.

Returns

*this

Returns `get(__s,__n,widen('\n'))`.

get() [6/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
             char_type * __s,
             streamsize __n,
             char_type __delim)
```

Simple multiple-character extraction.

Parameters

__s	Pointer to an array.
__n	Maximum number of characters to store in *__s*.
__delim	A "stop" character.

Returns

*this

Characters are extracted and stored into *__s* until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals *__delim*, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.
In any case, a null character is stored into the next location in the array.

Note

This function is not overloaded on signed char and unsigned char.

References _M_gcount, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_streambuf< _CharT, _Traits >:

getline() [1/3]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::getline (
             char_type * __s,
             streamsize __n)  [inline]
```

String extraction.

Parameters

_↩ _s	A character array in which to store the data.
_↩ _n	Maximum number of characters to extract.

Returns

*this

Returns `getline(__s,__n,widen('\n'))`.

getline() [2/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::getline (
            char_type * __s,
            streamsize __n,
            char_type __delim)
```

String extraction.

Parameters

__s	A character array in which to store the data.
__n	Maximum number of characters to extract.
__delim	A "stop" character.

Returns

*this

Extracts and stores characters into *__s* until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the *__s* array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)
In any case, a null character is stored in the next location in the array.
References _M_gcount, std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_streambuf< _CharT, _Traits >::sbumpc(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _Cha

getline() [3/3]

```
basic_istream< char > & std::basic_istream< char >::getline (
            char_type * __s,
            streamsize __n,
            char_type __delim)
```

Explicit specialization declarations, defined in src/istream.cc.

getloc()

```
locale std::ios_base::getloc () const  [inline], [nodiscard], [inherited]
```
Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::money_put< _CharT, _OutIter >::do_put(), std::basic_ios< _CharT, _Tr
std::chrono::operator<<(), std::operator>>(), std::operator>>(), and std::ws().

good()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good () const  [inline], [nodiscard], [inherited]
```
Fast error checking.

Returns

True if no error flags are set.

A wrapper around rdstate.
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::__detail::operator>>().

ignore() [1/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
            void )
```
Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().
References _M_gcount, std::ios_base::goodbit, and std::basic_ios< _CharT, _Traits >::rdbuf().

ignore() [2/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
            streamsize __n)
```
Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().
References _M_gcount, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_streambuf< _CharT, _Traits >:

ignore() [3/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
            streamsize __n,
            int_type __delim)
```

Discarding characters.

Parameters

__n	Number of characters to discard.
__delim	A "stop" character.

Returns

*this

Extracts characters and throws them away until one of the following happens:

- if *__n* != `std::numeric_limits<int>::max()`, *__n* characters are extracted
- the input sequence reaches end-of-file
- the next character equals *__delim* (in this case, the character is extracted); note that this condition will never occur if *__delim* equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.
References _M_gcount, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_streambuf< _CharT, _Traits >:

imbue()

```
template<typename _CharT, typename _Traits>
locale std::basic_ios< _CharT, _Traits >::imbue (
            const locale & __loc)  [inherited]
```

Moves to a new locale.

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.
Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html
References std::ios_base::getloc(), std::ios_base::imbue(), and rdbuf().
Referenced by std::chrono::operator<<().

init()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::init (
            basic_streambuf< _CharT, _Traits > * __sb)  [protected], [inherited]
```

All setup is performed here.
This is called from the public constructor. It is not virtual and cannot be redefined.
Referenced by std::basic_ios< char_type, traits_type >::basic_ios(), and std::basic_ostream< char_type, traits_type >::basic_ostream().

iword()

```
long & std::ios_base::iword (
            int __ix)  [inline], [inherited]
```

Access to integer array.

Parameters

_↩ _ix	Index into the array.

Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.
References iword().
Referenced by iword().

narrow()

```
template<typename _CharT, typename _Traits>
char std::basic_ios< _CharT, _Traits >::narrow (
            char_type __c,
            char __dfault) const  [inline], [inherited]
```

Squeezes characters.

Parameters

__c	The character to narrow.
__dfault	The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.
Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c,dfault)
```

Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.↩html

operator bool()

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool () const  [inline], [explicit], [nodiscard],
[inherited]
```

The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

operator"!()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator!  () const  [inline], [nodiscard], [inherited]
```

The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

operator>>() [1/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            __ios_type &(* __pf )(__ios_type &))  [inline]
```

Interface for manipulators.
Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`.
For more information, see the iomanip header.

operator>>() [2/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            __istream_type &(* __pf )(__istream_type &))  [inline]
```

Interface for manipulators.
Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`.
For more information, see the iomanip header.

operator>>() [3/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
            __streambuf_type * __sb)
```

Extracting into another streambuf.

Parameters

__sb	A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.
If `__sb` is NULL, the stream will set failbit in its error state.
Characters are extracted from this stream and inserted into the `__sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.
References std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

operator>>() [4/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            bool & __n)  [inline]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [5/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            double & __f)  [inline]
```

Floating point arithmetic extractors.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [6/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            float & __f)  [inline]
```

Floating point arithmetic extractors.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [7/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
            int & __n)
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.
References std::ios_base::badbit, std::ios_base::failbit, std::num_get< _CharT, _InIter >::get(), std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::setstate(), and std::use_facet().

operator>>() [8/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            ios_base &(* __pf )(ios_base &))  [inline]
```

Interface for manipulators.
Manipulators such as std::ws and std::dec use these functions in constructs like std::cin >> std::ws. For more information, see the iomanip header.

operator>>() [9/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            long & __n)  [inline]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [10/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            long double & __f)  [inline]
```

Floating point arithmetic extractors.

Parameters

↩ _↩ ↩ _↩ f	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [11/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            long long & __n)  [inline]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [12/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
            short & __n)
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.
References std::ios_base::badbit, std::ios_base::failbit, std::num_get< _CharT, _InIter >::get(), std::ios_base::goodbit, and std::use_facet().

operator>>() [13/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned int & __n)  [inline]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [14/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned long & __n)  [inline]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [15/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned long long & __n)  [inline]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [16/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned short & __n)  [inline]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [17/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            void *& __p)  [inline]
```

Basic arithmetic extractors.

Parameters

_↩ _p	A variable of pointer type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

peek()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::peek (
            void )
```

Looking ahead in the stream.

Returns

The next character, or eof().

If, after constructing the sentry object, good() is false, returns traits::eof(). Otherwise reads but does not extract the next input character.

precision() [1/2]

```
streamsize std::ios_base::precision () const  [inline], [nodiscard], [inherited]
```

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::chrono::operator<<().

precision() [2/2]

```
streamsize std::ios_base::precision (
            streamsize __prec)  [inline], [inherited]
```

Changing flags.

Parameters

__prec	The new precision value.

Returns

The previous value of precision().

putback()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::putback (
            char_type __c)
```

Unextracting a single character.

Parameters

_↩ _c	The character to push back into the input stream.

Returns

∗this

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.
If `rdbuf()` is null or if `sputbackc()` fails, sets badbit in the error state.

Note

This function first clears eofbit. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References _M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::rdstate(), std::basic_ios< _CharT, _Traits >::setstate(), and std::basic_streambuf< _CharT, _Traits >::sputbackc().

pword()

```
void *& std::ios_base::pword (
            int __ix)  [inline], [inherited]
```

Access to void pointer array.

Parameters

_↩ _ix	Index into the array.

Returns

A reference to a void∗ associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

rdbuf() [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf () const  [inline],
[nodiscard], [inherited]
```

Accessing the underlying buffer.

Returns

The current stream buffer.

This does not change the state of the stream.
Referenced by clear(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getlin std::getline(), std::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::ignore(), imbue(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::tr2::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >: std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits > std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::t std::basic_ostream< _CharT, _Traits >::tellp(), and std::ws().

rdbuf() [2/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
            basic_streambuf< _CharT, _Traits > * __sb)  [inherited]
```

Changing the underlying buffer.

Parameters

__sb	The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.
Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream     foo;         // or some other derived type
std::streambuf*  p = .....;

foo.ios::rdbuf(p);            // ios == basic_ios<char>
```

References clear().

rdstate()

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::rdstate () const  [inline], [nodiscard], [inherited]
```

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See std::ios_base::iostate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().
Referenced by std::basic_ios< char_type, traits_type >::bad(), clear(), std::basic_ios< char_type, traits_type >::eof(), std::basic_ios< char_type, traits_type >::fail(), std::basic_ios< char_type, traits_type >::good(), std::basic_istream< _CharT, _Traits >::| std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char_type, traits_type > and std::basic_istream< _CharT, _Traits >::unget().

read()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::read (
            char_type * __s,
            streamsize __n)
```

Extraction without delimiters.

Parameters

_↩ _s	A character array.
_↩ _n	Maximum number of characters to store.

Returns

*this

If the stream state is `good()`, extracts characters and stores them into *__s* until one of the following happens:

- *__n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

Note

This function is not overloaded on signed char and unsigned char.

References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

readsome()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::readsome (
            char_type * __s,
            streamsize __n)
```

Extraction until the buffer is exhausted, but no more.

Parameters

_↩ _s	A character array.
_↩ _n	Maximum number of characters to store.

Returns

The number of characters extracted.

Extracts characters and stores them into *__s* depending on the number of characters remaining in the streambuf's buffer, `rdbuf()->in_avail()`, called `A` here:

- if `A == -1`, sets eofbit and extracts no characters
- if `A == 0`, extracts no characters

- if A > 0, extracts min(A,n)

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.
References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::min(), std::basic_ios< _CharT, _Traits >::
and std::basic_ios< _CharT, _Traits >::setstate().

register_callback()

```
void std::ios_base::register_callback (
            event_callback __fn,
            int __index)  [inherited]
```

Add the callback __fn with parameter __index.

Parameters

__fn	The function to add.
__index	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.
References fmtflags.

seekg() [1/2]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
            off_type __off,
            ios_base::seekdir __dir)
```

Changing the current read position.

Parameters

__off	A file offset object.
__dir	The direction in which to seek.

Returns

*this

If fail() is not true, calls rdbuf()->pubseekoff(__off,__dir). If that function fails, sets failbit.

Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to gcount().

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::basic_ios< _CharT, _Traits >::fail(
std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::in, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits
and std::basic_ios< _CharT, _Traits >::setstate().

seekg() [2/2]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
            pos_type __pos)
```

Changing the current read position.

Parameters

__pos	A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets failbit.

Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::in, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::setstate() and std::basic_ios< _CharT, _Traits >::setstate().

setf() [1/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl)  [inline], [inherited]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.
References fmtflags.
Referenced by std::__detail::operator>>().

setf() [2/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl,
            fmtflags __mask)  [inline], [inherited]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.
__mask	The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.
References fmtflags.

setstate()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
            iostate __state)  [inline], [inherited]
```

Sets additional flags in the error state.

Parameters

__state	The additional state flag(s) to set.

See std::ios_base::iostate for the possible bit values.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::getline(), std::getline(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _Cha std::operator>>(), std::operator>>(), std::operator>>(), std::tr2::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >: std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits > std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), and std::ws().

sync()

```
template<typename _CharT, typename _Traits>
int std::basic_istream< _CharT, _Traits >::sync (
            void )
```

Synchronizing the stream buffer.

Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.
Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit and returns -1.
Otherwise, returns 0.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_streambuf< _CharT, _Traits >::pubsync(), std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

sync_with_stdio()

```
static bool std::ios_base::sync_with_stdio (
            bool __sync = true)  [static], [inherited]
```

Interaction with the standard C I/O objects.

Parameters

__sync	Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See https://gcc.gnu.↩org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary

tellg()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::pos_type std::basic_istream< _CharT, _Traits >::tellg (
            void )
```

Getting the current read position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type`(-1) to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0,cur,in)`.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with putback, unget and seekg, eofbit is not cleared first.

References std::ios_base::badbit, std::ios_base::cur, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::in, and std::basic_ios< _CharT, _Traits >::rdbuf().

tie() [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie () const  [inline],
[nodiscard], [inherited]
```

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and copyfmt().

tie() [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie (
            basic_ostream< _CharT, _Traits > * __tiestr)  [inline], [inherited]
```

Ties this stream to an output stream.

Parameters

__tiestr	The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

unget()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::unget (
            void )
```

Unextracting the previous character.

Returns

*this

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.
If `rdbuf()` is null or if `sungetc()` fails, sets badbit in the error state.

Note

This function first clears eofbit. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References _M_gcount, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::ios_base::goodbit, and std::basic_ios< _CharT, _Traits >::rdstate().
Referenced by std::__detail::operator>>().

unsetf()

```
void std::ios_base::unsetf (
             fmtflags __mask)  [inline], [inherited]
```

Clearing format flags.

Parameters

__mask	The flags to unset.

This function clears *__mask* in the format flags.
References fmtflags.

widen()

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
             char __c) const  [inline], [inherited]
```

Widens characters.

Parameters

_c	The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.
Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html
Referenced by std::basic_ios< char_type, traits_type >::fill(), std::getline(), std::getline(), and std::tr2::operator>>().

width() [1/2]

```
streamsize std::ios_base::width () const  [inline], [nodiscard], [inherited]
```

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_put< _CharT, _OutIter >::do_put(), std::operator>>(), and std::operator>>().

width() [2/2]

```
streamsize std::ios_base::width (
            streamsize __wide)  [inline], [inherited]
```

Changing flags.

Parameters

__wide	The new width value.

Returns

The previous value of width().

xalloc()

```
static int std::ios_base::xalloc () throw ( )   [static], [inherited]
```

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the iword and pword functions. The expectation is that an application calls xalloc in order to obtain an index in the iword and pword arrays that can be used without fear of conflict.
The implementation maintains a static variable that is incremented and returned on each invocation. xalloc is guaranteed to return an index that is safe to use in the iword and pword arrays.

5.225.6 Member Data Documentation

_M_gcount

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::_M_gcount  [protected]
```

The number of characters extracted in the previous unformatted function; see gcount().
Referenced by get(), get(), get(), get(), getline(), ignore(), ignore(), ignore(), putback(), read(), readsome(), and unget().

adjustfield

```
const fmtflags std::ios_base::adjustfield  [static], [inherited]
```

A mask of left|right|internal. Useful for the 2-arg form of `setf`.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::internal(), std::left(), and std::right().

app

```
const openmode std::ios_base::app  [static], [inherited]
```

Seek to end before each write.
Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

ate

const openmode std::ios_base::ate [static], [inherited]
Open and seek to end immediately after opening.
Referenced by std::basic_filebuf< _CharT, _Traits >::open().

badbit

const iostate std::ios_base::badbit [static], [inherited]
Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< char_type, traits_type >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::basic_ios< _CharT, _Traits >::clear(), std::basic_ostream< _CharT, _Trai std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_ostream< _CharT, _Traits >::opera std::basic_ostream< char_type, traits_type >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >: std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< char_type, traits_ty std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_istream< char_type, traits_type and std::ws().

basefield

const fmtflags std::ios_base::basefield [static], [inherited]
A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.
Referenced by std::dec(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::hex(), and std::oct().

beg

const seekdir std::ios_base::beg [static], [inherited]
Request a seek relative to the beginning of the stream.
Referenced by std::basic_filebuf< _CharT, _Traits >::seekpos().

binary

const openmode std::ios_base::binary [static], [inherited]
Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.↩ filestreams.binary.
Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

boolalpha

const fmtflags std::ios_base::boolalpha [static], [inherited]
Insert/extract `bool` in alphabetic rather than numeric format.
Referenced by std::boolalpha(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::noboolalpha().

cur

const seekdir std::ios_base::cur [static], [inherited]
Request a seek relative to the current position within the sequence.
Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

dec

const fmtflags std::ios_base::dec [static], [inherited]
Converts integer input or generates integer output in decimal base.
Referenced by std::dec().

end

const seekdir std::ios_base::end [static], [inherited]
Request a seek relative to the current end of the sequence.
Referenced by std::basic_filebuf< _CharT, _Traits >::open(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff().

eofbit

const iostate std::ios_base::eofbit [static], [inherited]
Indicates that an input operation reached the end of an input sequence.
Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >:
std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(),
std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter
std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::ge
std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< char_type, traits_type >::g
std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::putback(),
std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< char_type, traits_type >::read(), std::basic_istream< _CharT, _Traits
std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< char_type, traits_ty
std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

failbit

const iostate std::ios_base::failbit [static], [inherited]
Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::num_get< _CharT, _InIter >::do_get(),
std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(),
std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::ge
std::basic_istream< char_type, traits_type >::get(), std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::oper
std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(),
std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::s
std::basic_ostream< _CharT, _Traits >::seekp(), and std::basic_ostream< _CharT, _Traits >::seekp().

fixed

const fmtflags std::ios_base::fixed [static], [inherited]
Generate floating-point output in fixed-point notation.
Referenced by std::fixed(), and std::hexfloat().

floatfield

const fmtflags std::ios_base::floatfield [static], [inherited]
A mask of scientific|fixed. Useful for the 2-arg form of `setf`.
Referenced by std::defaultfloat(), std::fixed(), std::hexfloat(), and std::scientific().

goodbit

const iostate std::ios_base::goodbit [static], [inherited]
Indicates all is well.

Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::d std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >: std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore() std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ostream< _CharT, _Traits > std::basic_ostream< char_type, traits_type >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >:: std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits std::basic_istream< char_type, traits_type >::seekg(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Trait and std::ws().

hex

```
const fmtflags std::ios_base::hex  [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.
Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::hex().

in

```
const openmode std::ios_base::in  [static], [inherited]
```

Open for input. Default for `ifstream` and fstream.
Referenced by std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_stringbuf< _CharT, _Traits, _ std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_filebuf< _CharT, _Traits >::underflow(), std::basic_stringbuf< _CharT, _Traits, _ and std::basic_filebuf< _CharT, _Traits >::xsgetn().

internal

```
const fmtflags std::ios_base::internal  [static], [inherited]
```

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.
Referenced by std::internal().

left

```
const fmtflags std::ios_base::left  [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)
Referenced by std::num_put< _CharT, _OutIter >::do_put(), and std::left().

oct

```
const fmtflags std::ios_base::oct  [static], [inherited]
```

Converts integer input or generates integer output in octal base.
Referenced by std::oct().

out

```
const openmode std::ios_base::out  [static], [inherited]
```

Open for output. Default for `ofstream` and fstream.
Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(),

std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_stringbuf< _CharT, _Traits, std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

right

`const fmtflags std::ios_base::right [static], [inherited]`
Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)
Referenced by std::right().

scientific

`const fmtflags std::ios_base::scientific [static], [inherited]`
Generates floating-point output in scientific notation.
Referenced by std::hexfloat(), and std::scientific().

showbase

`const fmtflags std::ios_base::showbase [static], [inherited]`
Generates a prefix indicating the numeric base of generated integer output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::noshowbase(), and std::showbase().

showpoint

`const fmtflags std::ios_base::showpoint [static], [inherited]`
Generates a decimal-point character unconditionally in generated floating-point output.
Referenced by std::noshowpoint(), and std::showpoint().

showpos

`const fmtflags std::ios_base::showpos [static], [inherited]`
Generates a + sign in non-negative generated numeric output.
Referenced by std::noshowpos(), and std::showpos().

skipws

`const fmtflags std::ios_base::skipws [static], [inherited]`
Skips leading white space before certain input operations.
Referenced by std::noskipws(), std::__detail::operator>>(), and std::skipws().

trunc

`const openmode std::ios_base::trunc [static], [inherited]`
Truncate an existing stream when opening. Default for `ofstream`.

unitbuf

`const fmtflags std::ios_base::unitbuf [static], [inherited]`
Flushes output after each output operation.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::nounitbuf(), and std::unitbuf().

uppercase

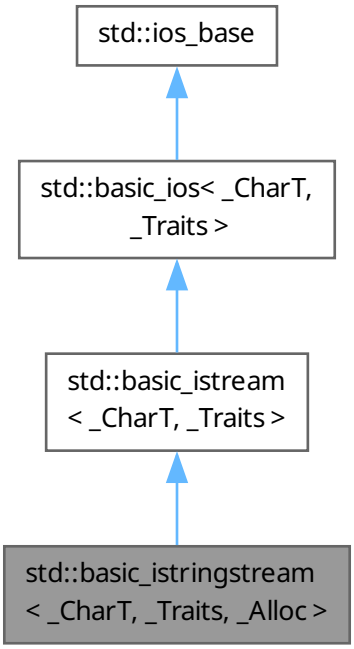
`const fmtflags std::ios_base::uppercase [static], [inherited]`
Replaces certain lowercase letters with their uppercase equivalents in generated output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::nouppercase(), and std::uppercase().
The documentation for this class was generated from the following files:

- iosfwd
- istream
- istream.tcc

5.226 std::basic_istringstream< _CharT, _Traits, _Alloc > Class Template Reference

`#include <sstream>`

Inheritance diagram for std::basic_istringstream< _CharT, _Traits, _Alloc >:



Public Types

- typedef ctype< _CharT > **__ctype_type**
- typedef basic_ios< _CharT, _Traits > **__ios_type**
- typedef basic_istream< char_type, traits_type > **__istream_type**
- typedef num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > **__num_get_type**
- typedef basic_streambuf< _CharT, _Traits > **__streambuf_type**
- typedef basic_string< _CharT, _Traits, _Alloc > **__string_type**
- typedef basic_stringbuf< _CharT, _Traits, _Alloc > **__stringbuf_type**
- typedef _Alloc **allocator_type**
- typedef _CharT **char_type**
- enum event { **erase_event** , **imbue_event** , **copyfmt_event** }
- typedef void(* event_callback) (event __e, ios_base &__b, int __i)
- typedef traits_type::int_type **int_type**
- typedef _Ios_Iostate iostate
- typedef traits_type::off_type **off_type**

- typedef _Ios_Openmode openmode
- typedef traits_type::pos_type **pos_type**
- typedef _Ios_Seekdir seekdir
- typedef _Traits **traits_type**

- typedef num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > __num_put_type

Public Member Functions

- basic_istringstream ()
- **basic_istringstream** (__string_type &&__str, ios_base::openmode __mode=ios_base::in)
- **basic_istringstream** (basic_istringstream &&__rhs)
- basic_istringstream (const __string_type &__str, ios_base::openmode __mode=ios_base::in)
- **basic_istringstream** (const basic_istringstream &)=delete
- template<typename _SAlloc>
  basic_istringstream (const basic_string< _CharT, _Traits, _SAlloc > &__str, const allocator_type &__a)
- template<typename _SAlloc>
  basic_istringstream (const basic_string< _CharT, _Traits, _SAlloc > &__str, ios_base::openmode __mode, const allocator_type &__a)
- template<typename _SAlloc>
  basic_istringstream (const basic_string< _CharT, _Traits, _SAlloc > &__str, ios_base::openmode __↩ mode=ios_base::in)
- basic_istringstream (ios_base::openmode __mode)
- **basic_istringstream** (ios_base::openmode __mode, const allocator_type &__a)
- ~basic_istringstream ()
- template<typename _ValueT>
  basic_istream< _CharT, _Traits > & **_M_extract** (_ValueT &__v)
- const locale & _M_getloc () const
- void **_M_setstate** (iostate __state)
- bool bad () const
- void clear (iostate __state=goodbit)
- basic_ios & copyfmt (const basic_ios &__rhs)
- bool eof () const
- iostate exceptions () const
- void exceptions (iostate __except)
- bool fail () const
- char_type fill () const
- char_type fill (char_type __ch)
- fmtflags flags () const
- fmtflags flags (fmtflags __fmtfl)
- streamsize gcount () const
- basic_istream< char > & getline (char_type *__s, streamsize __n, char_type __delim)
- basic_istream< wchar_t > & **getline** (char_type *__s, streamsize __n, char_type __delim)
- locale getloc () const
- bool good () const
- basic_istream< char > & **ignore** (streamsize __n)
- basic_istream< wchar_t > & **ignore** (streamsize __n)
- basic_istream< char > & **ignore** (streamsize __n, int_type __delim)
- basic_istream< wchar_t > & **ignore** (streamsize __n, int_type __delim)
- locale imbue (const locale &__loc)
- long & iword (int __ix)

- char narrow (char_type __c, char __dfault) const
- basic_istringstream & **operator=** (basic_istringstream &&__rhs)
- basic_istringstream & **operator=** (const basic_istringstream &)=delete
- __istream_type & operator>> (__streambuf_type *__sb)
- __istream_type & operator>> (void *&__p)
- streamsize precision () const
- streamsize precision (streamsize __prec)
- void *& pword (int __ix)
- basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > *__sb)
- __stringbuf_type * rdbuf () const
- iostate rdstate () const
- void register_callback (event_callback __fn, int __index)
- fmtflags setf (fmtflags __fmtfl)
- fmtflags setf (fmtflags __fmtfl, fmtflags __mask)
- void setstate (iostate __state)
- __string_type **str** () &&
- __string_type str () const &
- void **str** (__string_type &&__s)
- void str (const __string_type &__s)
- template<__allocator_like _SAlloc>
  basic_string< _CharT, _Traits, _SAlloc > **str** (const _SAlloc &__sa) const
- template<__allocator_like _SAlloc>
  requires (!is_same_v<_SAlloc, _Alloc>)
  void **str** (const basic_string< _CharT, _Traits, _SAlloc > &__s)
- void **swap** (basic_istringstream &__rhs)
- basic_ostream< _CharT, _Traits > * tie () const
- basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > *__tiestr)
- void unsetf (fmtflags __mask)
- basic_string_view< char_type, traits_type > **view** () const noexcept
- char_type widen (char __c) const
- streamsize width () const
- streamsize width (streamsize __wide)

- __istream_type & operator>> (__istream_type &(*__pf)(__istream_type &))
- __istream_type & operator>> (__ios_type &(*__pf)(__ios_type &))
- __istream_type & operator>> (ios_base &(*__pf)(ios_base &))

Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type std::basic_istream::sentry with the second argument (noskipws) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, ios_base::badbit will be turned on in the stream's error state (without causing an ios_base::failure to be thrown) and the original exception will be rethrown if badbit is set in the exceptions mask.

- __istream_type & operator>> (bool &__n)
- __istream_type & operator>> (short &__n)
- __istream_type & operator>> (unsigned short &__n)
- __istream_type & operator>> (int &__n)
- __istream_type & operator>> (unsigned int &__n)
- __istream_type & operator>> (long &__n)

- __istream_type & operator>> (unsigned long &__n)
- __istream_type & operator>> (long long &__n)
- __istream_type & operator>> (unsigned long long &__n)

- __istream_type & operator>> (float &__f)
- __istream_type & operator>> (double &__f)
- __istream_type & operator>> (long double &__f)

Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type std::basic_istream::sentry with the second argument (noskipws) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by gcount().

If an exception is thrown during extraction, ios_base::badbit will be turned on in the stream's error state (without causing an ios_base::failure to be thrown) and the original exception will be rethrown if badbit is set in the exceptions mask.

- int_type get ()
- __istream_type & get (char_type &__c)
- __istream_type & get (char_type *__s, streamsize __n, char_type __delim)
- __istream_type & get (char_type *__s, streamsize __n)
- __istream_type & get (__streambuf_type &__sb, char_type __delim)
- __istream_type & get (__streambuf_type &__sb)
- __istream_type & getline (char_type *__s, streamsize __n, char_type __delim)
- __istream_type & getline (char_type *__s, streamsize __n)
- __istream_type & ignore (streamsize __n, int_type __delim)
- __istream_type & ignore (streamsize __n)
- __istream_type & ignore ()
- int_type peek ()
- __istream_type & read (char_type *__s, streamsize __n)
- streamsize readsome (char_type *__s, streamsize __n)
- __istream_type & putback (char_type __c)
- __istream_type & unget ()
- int sync ()
- pos_type tellg ()
- __istream_type & seekg (pos_type)
- __istream_type & seekg (off_type, ios_base::seekdir)

- operator bool () const
- bool operator! () const

Static Public Member Functions

- static bool sync_with_stdio (bool __sync=true)
- static int xalloc () throw ()

Public Attributes

- class __attribute((__abi_tag__("cxx11"))) failure typedef _Ios_Fmtflags fmtflags

Static Public Attributes

- static const openmode **__noreplace**
- static const fmtflags adjustfield
- static const openmode app
- static const openmode ate
- static const iostate badbit
- static const fmtflags basefield
- static const seekdir beg
- static const openmode binary
- static const fmtflags boolalpha
- static const seekdir cur
- static const fmtflags dec
- static const seekdir end
- static const iostate eofbit
- static const iostate failbit
- static const fmtflags fixed
- static const fmtflags floatfield
- static const iostate goodbit
- static const fmtflags hex
- static const openmode in
- static const fmtflags internal
- static const fmtflags left
- static const fmtflags oct
- static const openmode out
- static const fmtflags right
- static const fmtflags scientific
- static const fmtflags showbase
- static const fmtflags showpoint
- static const fmtflags showpos
- static const fmtflags skipws
- static const openmode trunc
- static const fmtflags unitbuf
- static const fmtflags uppercase

Protected Types

- enum { **_S_local_word_size** }

Protected Member Functions

- void **_M_cache_locale** (const locale &__loc)
- void **_M_call_callbacks** (event __ev) throw ()
- void **_M_dispose_callbacks** (void) throw ()
- template<typename _ValueT>
  __istream_type & **_M_extract** (_ValueT &__v)
- _Words & **_M_grow_words** (int __index, bool __iword)
- void **_M_init** () throw ()
- void **_M_move** (ios_base &) noexcept
- void **_M_swap** (ios_base &__rhs) noexcept
- void init (basic_streambuf< _CharT, _Traits > *__sb)
- void **move** (basic_ios &&__rhs)

- void **move** (basic_ios &__rhs)
- void **set_rdbuf** (basic_streambuf< _CharT, _Traits > *__sb)
- void **swap** (basic_ios &__rhs) noexcept
- void **swap** (basic_istream &__rhs)

Protected Attributes

- _Callback_list * **_M_callbacks**
- const __ctype_type * **_M_ctype**
- iostate **_M_exception**
- char_type **_M_fill**
- bool **_M_fill_init**
- fmtflags **_M_flags**
- streamsize _M_gcount
- locale **_M_ios_locale**
- _Words **_M_local_word** [_S_local_word_size]
- const __num_get_type * **_M_num_get**
- const __num_put_type * **_M_num_put**
- streamsize **_M_precision**
- basic_streambuf< _CharT, _Traits > * **_M_streambuf**
- iostate **_M_streambuf_state**
- basic_ostream< _CharT, _Traits > * **_M_tie**
- streamsize **_M_width**
- _Words * **_M_word**
- int **_M_word_size**
- _Words **_M_word_zero**

5.226.1 Detailed Description

template<typename _CharT, typename _Traits, typename _Alloc>
class std::basic_istringstream< _CharT, _Traits, _Alloc >

Controlling input for std::string.

Template Parameters

_CharT	Type of character stream.
_Traits	Traits for character type, defaults to char_traits<_CharT>.
_Alloc	Allocator type, defaults to allocator<_CharT>.

This class supports reading from objects of type std::basic_string, using the inherited functions from std::basic_istream. To control the associated sequence, an instance of std::basic_stringbuf is used, which this page refers to as `sb`.

5.226.2 Member Typedef Documentation

__num_put_type

```
template<typename _CharT, typename _Traits>
typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios< _CharT, _Traits
>::__num_put_type  [inherited]
```

These are non-standard types.

event_callback

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i)  [inherited]
```

The type of an event callback function.

Parameters

_↩ _e	One of the members of the event enum.
_↩ _b	Reference to the ios_base object.
_↩ _i	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several ios_base and basic_ios functions, specifically imbue(), copyfmt(), and ~ios().

iostate

`typedef _Ios_Iostate std::ios_base::iostate [inherited]`
This is a bitmask type.
_Ios_Iostate is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit
- eofbit
- failbit
- goodbit

openmode

`typedef _Ios_Openmode std::ios_base::openmode [inherited]`
This is a bitmask type.
_Ios_Openmode is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type openmode are:

- app
- ate
- binary
- in
- out
- trunc

seekdir

`typedef _Ios_Seekdir std::ios_base::seekdir [inherited]`
This is an enumerated type.
_Ios_Seekdir is implementation-defined. Defined values of type seekdir are:

- beg
- cur, equivalent to SEEK_CUR in the C standard library.
- end, equivalent to SEEK_END in the C standard library.

5.226.3 Member Enumeration Documentation

event

```
enum std::ios_base::event  [inherited]
```

The set of events that may be passed to an event callback.
erase_event is used during ~ios() and copyfmt(). imbue_event is used during imbue(). copyfmt_event is used during copyfmt().

5.226.4 Constructor & Destructor Documentation

basic_istringstream() [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_istringstream< _CharT, _Traits, _Alloc >::basic_istringstream ()  [inline]
```

Default constructor starts with an empty string buffer.
Initializes `sb` using `in`, and passes `&sb` to the base class initializer. Does not allocate any buffer.
That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

basic_istringstream() [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_istringstream< _CharT, _Traits, _Alloc >::basic_istringstream (
            ios_base::openmode __mode)  [inline], [explicit]
```

Starts with an empty string buffer.

Parameters

__mode	Whether the buffer can read, or write, or both.

`ios_base::in` is automatically included in *__mode*.
Initializes `sb` using `__mode|in`, and passes `&sb` to the base class initializer. Does not allocate any buffer.
That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

basic_istringstream() [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_istringstream< _CharT, _Traits, _Alloc >::basic_istringstream (
            const __string_type & __str,
            ios_base::openmode __mode = ios_base::in)  [inline], [explicit]
```

Starts with an existing string buffer.

Parameters

__str	A string to copy as a starting buffer.
__mode	Whether the buffer can read, or write, or both.

`ios_base::in` is automatically included in *mode*.
Initializes `sb` using *str* and `mode|in`, and passes `&sb` to the base class initializer.
That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

~basic_istringstream()

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_istringstream< _CharT, _Traits, _Alloc >::~basic_istringstream ()  [inline]
```

The destructor does nothing.
The buffer is deallocated by the stringbuf object, not the formatting stream.

5.226.5 Member Function Documentation

_M_getloc()

```
const locale & std::ios_base::_M_getloc () const  [inline], [inherited]
```

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.
Referenced by std::money_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter
std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIte
std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::num_put< _CharT, _OutIter
std::time_put< _CharT, _OutIter >::do_put(), std::time_get< _CharT, _InIter >::get(), and std::time_put< _CharT, _OutIter >::put().

bad()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry().

clear()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::clear (
            iostate __state = goodbit)  [inherited]
```

[Re]sets the error state.

Parameters

__state	The new state flag(s) to set.

See std::ios_base::iostate for the possible bit values. Most users will not need to pass an argument.
References std::ios_base::badbit, exceptions(), rdbuf(), and rdstate().
Referenced by std::basic_ios< char_type, traits_type >::exceptions(), std::__detail::operator>>(), std::basic_istream< _CharT, _Traits >
rdbuf(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char_type, traits
and std::basic_istream< _CharT, _Traits >::unget().

copyfmt()

```
template<typename _CharT, typename _Traits>
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
            const basic_ios< _CharT, _Traits > & __rhs)  [inherited]
```

Copies fields of __rhs into this.

Parameters

__rhs	The source values for the copies.

Returns

Reference to this object.

All fields of __rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase_event. After copying, each (new) callback is invoked with copyfmt_event. The final step is to copy exceptions().
References basic_ios(), std::__addressof(), exceptions(), fill(), std::ios_base::flags(), std::ios_base::getloc(), std::ios_base::precision(), tie(), std::tie(), and std::ios_base::width().

eof()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

exceptions() [1/2]

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions () const  [inline], [nodiscard], [inherited]
```

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.
Referenced by clear(), and copyfmt().

exceptions() [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
            iostate __except)  [inline], [inherited]
```

Throwing exceptions on errors.

Parameters

__except	The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type std::ios_base::failure is thrown.
If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr « "Setting badbit\n";
```

```
    f.setstate (std::ios_base::badbit);

    std::cerr « "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.
Referenced by std::basic_ios< char_type, traits_type >::operator bool(), std::basic_ios< char_type, traits_type >::operator!(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >:: std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >:: and std::regex_traits< _Ch_type >::value().

fill() [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const  [inline], [nodiscard], [inherited]
```

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.
Referenced by copyfmt().

fill() [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
             char_type __ch)  [inline], [inherited]
```

Sets a new *empty* character.

Parameters

__ch	The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

flags() [1/2]

```
fmtflags std::ios_base::flags () const  [inline], [nodiscard], [inherited]
```

Access to format flags.

Returns

The format control flags for both input and output.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >
std::num_put< _CharT, _OutIter >::do_put(), std::num_put< _CharT, _OutIter >::do_put(), std::chrono::operator<<(),
std::operator<<(), std::__detail::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(),
std::operator>>(), std::operator>>(), and std::operator>>().

flags() [2/2]

```
fmtflags std::ios_base::flags (
            fmtflags __fmtfl)  [inline], [inherited]
```

Setting new format flags all at once.

Parameters

__fmtfl	The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *__fmtfl*.
References fmtflags.

gcount()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::gcount () const  [inline], [inherited]
```

Character counting.

Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

get() [1/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::get (
            void )  [inherited]
```

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().
References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit,
std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

get() [2/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::get (
            __streambuf_type & __sb)  [inline], [inherited]
```

Extraction into another streambuf.

Parameters

__sb	A streambuf in which to store data.

Returns

*this

Returns `get(__sb,widen('\n'))`.

get() [3/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
            __streambuf_type & __sb,
            char_type __delim)  [inherited]
```

Extraction into another streambuf.

Parameters

__sb	A streambuf in which to store data.
__delim	A "stop" character.

Returns

*this

Characters are extracted and inserted into *__sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals *__delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.
References _M_gcount, and std::ios_base::goodbit.

get() [4/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
            char_type & __c)  [inherited]
```

Simple extraction.

Parameters

__c	The character in which to store data.

Returns

*this

Tries to extract a character and store it in *__c*. If none are available, sets failbit and returns traits::eof().

Note

This function is not overloaded on signed char and unsigned char.

References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

get() [5/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::get (
            char_type * __s,
            streamsize __n)  [inline], [inherited]
```

Simple multiple-character extraction.

Parameters

_↩ _s	Pointer to an array.
_↩ _n	Maximum number of characters to store in *s*.

Returns

*this

Returns `get(__s,__n,widen('\n'))`.

get() [6/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
            char_type * __s,
            streamsize __n,
            char_type __delim)  [inherited]
```

Simple multiple-character extraction.

Parameters

__s	Pointer to an array.
__n	Maximum number of characters to store in *__s*.
__delim	A "stop" character.

Returns

*this

Characters are extracted and stored into *__s* until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals *__delim*, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.
In any case, a null character is stored into the next location in the array.

Note

This function is not overloaded on signed char and unsigned char.

References _M_gcount, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_streambuf< _CharT, _Traits >:

getline() [1/3]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::getline (
            char_type * __s,
            streamsize __n)  [inline], [inherited]
```

String extraction.

Parameters

_↩ _s	A character array in which to store the data.
_↩ _n	Maximum number of characters to extract.

Returns

*this

Returns `getline(__s,__n,widen('\n'))`.

getline() [2/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::getline (
            char_type * __s,
            streamsize __n,
            char_type __delim)  [inherited]
```

String extraction.

Parameters

__s	A character array in which to store the data.
__n	Maximum number of characters to extract.
__delim	A "stop" character.

Returns

*this

Extracts and stores characters into __s until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the __s array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)
In any case, a null character is stored in the next location in the array.
References _M_gcount, std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_streambuf< _CharT, _Traits >::sbumpc(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _Cha

getline() [3/3]

```
basic_istream< char > & std::basic_istream< char >::getline (
            char_type * __s,
            streamsize __n,
            char_type __delim)  [inherited]
```

Explicit specialization declarations, defined in src/istream.cc.

getloc()

```
locale std::ios_base::getloc () const  [inline], [nodiscard], [inherited]
```

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::money_put< _CharT, _OutIter >::do_put(), std::basic_ios< _CharT, _Tr std::chrono::operator<<(), std::operator>>(), std::operator>>(), and std::ws().

good()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if no error flags are set.

A wrapper around rdstate.

Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::__detail::operator>>().

ignore() [1/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
            void )  [inherited]
```

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

References _M_gcount, std::ios_base::goodbit, and std::basic_ios< _CharT, _Traits >::rdbuf().

ignore() [2/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
            streamsize __n)  [inherited]
```

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

References _M_gcount, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_streambuf< _CharT, _Traits >:

ignore() [3/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
            streamsize __n,
            int_type __delim)  [inherited]
```

Discarding characters.

Parameters

__n	Number of characters to discard.
__delim	A "stop" character.

Returns

*this

Extracts characters and throws them away until one of the following happens:

- if *__n* != `std::numeric_limits<int>::max()`, *__n* characters are extracted
- the input sequence reaches end-of-file
- the next character equals *__delim* (in this case, the character is extracted); note that this condition will never occur if *__delim* equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.
References _M_gcount, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_streambuf< _CharT, _Traits >:

imbue()

```
template<typename _CharT, typename _Traits>
locale std::basic_ios< _CharT, _Traits >::imbue (
            const locale & __loc)  [inherited]
```

Moves to a new locale.

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.
Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.↩
html
References std::ios_base::getloc(), std::ios_base::imbue(), and rdbuf().
Referenced by std::chrono::operator<<().

init()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::init (
            basic_streambuf< _CharT, _Traits > * __sb)  [protected], [inherited]
```

All setup is performed here.
This is called from the public constructor. It is not virtual and cannot be redefined.
Referenced by std::basic_ios< char_type, traits_type >::basic_ios(), and std::basic_ostream< char_type, traits_type >::basic_ostream().

iword()

```
long & std::ios_base::iword (
            int __ix)  [inline], [inherited]
```

Access to integer array.

Parameters

_↩ _ix	Index into the array.

Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.
References iword().
Referenced by iword().

narrow()

```
template<typename _CharT, typename _Traits>
char std::basic_ios< _CharT, _Traits >::narrow (
            char_type __c,
            char __dfault) const  [inline], [inherited]
```

Squeezes characters.

Parameters

__c	The character to narrow.
__dfault	The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.
Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c,dfault)
```

Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.↩ html

operator bool()

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool () const  [inline], [explicit], [nodiscard],
[inherited]
```

The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

operator"!()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator!  () const  [inline], [nodiscard], [inherited]
```

The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

operator>>() [1/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            __ios_type &(* __pf )(__ios_type &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`.
For more information, see the iomanip header.

operator>>() [2/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            __istream_type &(* __pf )(__istream_type &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`.
For more information, see the iomanip header.

operator>>() [3/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
            __streambuf_type * __sb)  [inherited]
```

Extracting into another streambuf.

Parameters

__sb	A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.
If `__sb` is NULL, the stream will set failbit in its error state.
Characters are extracted from this stream and inserted into the `__sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.
References std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

operator>>() [4/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            bool & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [5/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            double & __f)  [inline], [inherited]
```

Floating point arithmetic extractors.

Parameters

↩ _↩ ↩ _↩ f	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [6/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            float & __f)  [inline], [inherited]
```

Floating point arithmetic extractors.

Parameters

↩ _↩ ↩ _↩ f	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [7/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
            int & __n)  [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.
References std::ios_base::badbit, std::ios_base::failbit, std::num_get< _CharT, _InIter >::get(), std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::setstate(), and std::use_facet().

operator>>() [8/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            ios_base &(* __pf )(ios_base &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as std::ws and std::dec use these functions in constructs like std::cin >> std::ws. For more information, see the iomanip header.

operator>>() [9/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            long & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [10/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            long double & __f)  [inline], [inherited]
```

Floating point arithmetic extractors.

Parameters

↩ _↩ ↩ _↩ f	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [11/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            long long & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [12/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
            short & __n)  [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.
References std::ios_base::badbit, std::ios_base::failbit, std::num_get< _CharT, _InIter >::get(), std::ios_base::goodbit, and std::use_facet().

operator>>() [13/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned int & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [14/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned long & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [15/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned long long & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [16/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned short & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [17/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            void *& __p)  [inline], [inherited]
```

Basic arithmetic extractors.

Parameters

_↩ _p	A variable of pointer type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

peek()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::peek (
            void )  [inherited]
```

Looking ahead in the stream.

Returns

The next character, or eof().

If, after constructing the sentry object, good() is false, returns traits::eof(). Otherwise reads but does not extract the next input character.

precision() [1/2]

```
streamsize std::ios_base::precision () const  [inline], [nodiscard], [inherited]
```

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::chrono::operator<<().

precision() [2/2]

```
streamsize std::ios_base::precision (
            streamsize __prec)  [inline], [inherited]
```

Changing flags.

Parameters

__prec	The new precision value.

Returns

The previous value of precision().

putback()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::putback (
            char_type __c)  [inherited]
```

Unextracting a single character.

Parameters

_↩ _c	The character to push back into the input stream.

Returns

∗this

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.
If `rdbuf()` is null or if `sputbackc()` fails, sets badbit in the error state.

Note

This function first clears eofbit. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References _M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::rdstate(), std::basic_ios< _CharT, _Traits >::setstate(), and std::basic_streambuf< _CharT, _Traits >::sputbackc().

pword()

```
void *& std::ios_base::pword (
            int __ix)  [inline], [inherited]
```

Access to void pointer array.

Parameters

_↩ _ix	Index into the array.

Returns

A reference to a void∗ associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

rdbuf() [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
            basic_streambuf< _CharT, _Traits > * __sb)  [inherited]
```

Changing the underlying buffer.

Parameters

__sb	The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.
Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream     foo;         // or some other derived type
std::streambuf*  p = .....;

foo.ios::rdbuf(p);            // ios == basic_ios<char>
```

References clear().

rdbuf() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
__stringbuf_type * std::basic_istringstream< _CharT, _Traits, _Alloc >::rdbuf () const  [inline],
[nodiscard]
```

Accessing the underlying buffer.

Returns

The current basic_stringbuf buffer.

This hides both signatures of std::basic_ios::rdbuf().

rdstate()

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::rdstate () const  [inline], [nodiscard], [inherited]
```

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See std::ios_base::iostate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().
Referenced by std::basic_ios< char_type, traits_type >::bad(), clear(), std::basic_ios< char_type, traits_type >::eof(), std::basic_ios< char_type, traits_type >::fail(), std::basic_ios< char_type, traits_type >::good(), std::basic_istream< _CharT, _Traits >::
std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char_type, traits_type >
and std::basic_istream< _CharT, _Traits >::unget().

read()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::read (
            char_type * __s,
            streamsize __n)  [inherited]
```

Extraction without delimiters.

Parameters

__s	A character array.

_↩ _n	Maximum number of characters to store.

Returns

*this

If the stream state is `good()`, extracts characters and stores them into __*s* until one of the following happens:

- __*n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

Note

This function is not overloaded on signed char and unsigned char.

References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

readsome()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::readsome (
            char_type * __s,
            streamsize __n)  [inherited]
```

Extraction until the buffer is exhausted, but no more.

Parameters

_↩ _s	A character array.
_↩ _n	Maximum number of characters to store.

Returns

The number of characters extracted.

Extracts characters and stores them into __*s* depending on the number of characters remaining in the streambuf's buffer, `rdbuf()->in_avail()`, called `A` here:

- if `A == -1`, sets eofbit and extracts no characters
- if `A == 0`, extracts no characters
- if `A > 0`, extracts `min(A,n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.
References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::min(), std::basic_ios< _CharT, _Traits >::
and std::basic_ios< _CharT, _Traits >::setstate().

register_callback()

```
void std::ios_base::register_callback (
            event_callback __fn,
            int __index)  [inherited]
```

Add the callback __fn with parameter __index.

Parameters

__fn	The function to add.
__index	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered. References fmtflags.

seekg() [1/2]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
            off_type __off,
            ios_base::seekdir __dir)  [inherited]
```

Changing the current read position.

Parameters

__off	A file offset object.
__dir	The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(__off,__dir)`. If that function fails, sets failbit.

Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::basic_ios< _CharT, _Traits >::fail(
std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::in, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits
and std::basic_ios< _CharT, _Traits >::setstate().

seekg() [2/2]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
            pos_type __pos)  [inherited]
```

Changing the current read position.

Parameters

__pos	A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets failbit.

Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::in, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::setstate(), and std::basic_ios< _CharT, _Traits >::setstate().

setf() [1/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl)  [inline], [inherited]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.
References fmtflags.
Referenced by std::__detail::operator>>().

setf() [2/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl,
            fmtflags __mask)  [inline], [inherited]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.
__mask	The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.
References fmtflags.

setstate()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
            iostate __state)  [inline], [inherited]
```

Sets additional flags in the error state.

Parameters

__state	The additional state flag(s) to set.

See std::ios_base::iostate for the possible bit values.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::getline(), std::getline(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _Cha std::operator>>(), std::operator>>(), std::operator>>(), std::tr2::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >: std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits > std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), and std::ws().

str() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
__string_type std::basic_istringstream< _CharT, _Traits, _Alloc >::str () const &  [inline],
[nodiscard]
```

Copying out the string buffer.

Returns

`rdbuf()->str()`

str() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_istringstream< _CharT, _Traits, _Alloc >::str (
            const __string_type & __s)  [inline]
```

Setting a new buffer.

Parameters

_↩ _s	The string to use as a new sequence.

Calls `rdbuf()->str(s)`.

sync()

```
template<typename _CharT, typename _Traits>
int std::basic_istream< _CharT, _Traits >::sync (
            void )  [inherited]
```

Synchronizing the stream buffer.

Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.
Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit and returns -1.
Otherwise, returns 0.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_streambuf< _CharT, _Traits >::pubsync(), std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

sync_with_stdio()

```
static bool std::ios_base::sync_with_stdio (
            bool __sync = true)  [static], [inherited]
```

Interaction with the standard C I/O objects.

Parameters

__sync	Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary

tellg()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::pos_type std::basic_istream< _CharT, _Traits >::tellg (
            void )  [inherited]
```

Getting the current read position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type`(-1) to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0,cur,in)`.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with putback, unget and seekg, eofbit is not cleared first.

References std::ios_base::badbit, std::ios_base::cur, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::in, and std::basic_ios< _CharT, _Traits >::rdbuf().

tie() [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie () const  [inline],
[nodiscard], [inherited]
```

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and copyfmt().

tie() [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie (
            basic_ostream< _CharT, _Traits > * __tiestr)  [inline], [inherited]
```

Ties this stream to an output stream.

Parameters

__tiestr	The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

unget()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::unget (
            void )  [inherited]
```

Unextracting the previous character.

Returns

*this

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.
If `rdbuf()` is null or if `sungetc()` fails, sets badbit in the error state.

Note

This function first clears eofbit. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References _M_gcount, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::ios_base::goodbit, and std::basic_ios< _CharT, _Traits >::rdstate().
Referenced by std::__detail::operator>>().

unsetf()

```
void std::ios_base::unsetf (
            fmtflags __mask)  [inline], [inherited]
```

Clearing format flags.

Parameters

__mask	The flags to unset.

This function clears *__mask* in the format flags.
References fmtflags.

widen()

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
            char __c) const  [inline], [inherited]
```

Widens characters.

Parameters

_↩ _c	The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.
Returns the result of
`std::use_facet<ctype<char_type> >(getloc()).widen(c)`
Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html
Referenced by std::basic_ios< char_type, traits_type >::fill(), std::getline(), std::getline(), and std::tr2::operator>>().

width() [1/2]

```
streamsize std::ios_base::width () const  [inline], [nodiscard], [inherited]
```
Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_put< _CharT, _OutIter >::do_put(), std::operator>>(), and std::operator>>().

width() [2/2]

```
streamsize std::ios_base::width (
             streamsize __wide)  [inline], [inherited]
```
Changing flags.

Parameters

__wide	The new width value.

Returns

The previous value of width().

xalloc()

```
static int std::ios_base::xalloc () throw ( )  [static], [inherited]
```
Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the iword and pword functions. The expectation is that an application calls xalloc in order to obtain an index in the iword and pword arrays that can be used without fear of conflict.
The implementation maintains a static variable that is incremented and returned on each invocation. xalloc is guaranteed to return an index that is safe to use in the iword and pword arrays.

5.226.6 Member Data Documentation

_M_gcount

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::_M_gcount  [protected], [inherited]
```
The number of characters extracted in the previous unformatted function; see gcount().
Referenced by get(), get(), get(), get(), getline(), ignore(), ignore(), ignore(), putback(), read(), readsome(), and unget().

adjustfield

```
const fmtflags std::ios_base::adjustfield  [static], [inherited]
```
A mask of left|right|internal. Useful for the 2-arg form of `setf`.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::internal(), std::left(), and std::right().

app

```
const openmode std::ios_base::app  [static], [inherited]
```
Seek to end before each write.
Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

ate

```
const openmode std::ios_base::ate  [static], [inherited]
```
Open and seek to end immediately after opening.
Referenced by std::basic_filebuf< _CharT, _Traits >::open().

badbit

```
const iostate std::ios_base::badbit  [static], [inherited]
```
Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< char_type, traits_type >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::basic_ios< _CharT, _Traits >::clear(), std::basic_ostream< _CharT, _Trai std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_ostream< _CharT, _Traits >::opera std::basic_ostream< char_type, traits_type >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >: std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< char_type, traits_ty std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_istream< char_type, traits_type and std::ws().

basefield

```
const fmtflags std::ios_base::basefield  [static], [inherited]
```
A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.
Referenced by std::dec(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::hex(), and std::oct().

beg

```
const seekdir std::ios_base::beg  [static], [inherited]
```
Request a seek relative to the beginning of the stream.
Referenced by std::basic_filebuf< _CharT, _Traits >::seekpos().

binary

```
const openmode std::ios_base::binary  [static], [inherited]
```
Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.↩filestreams.binary.
Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

boolalpha

```
const fmtflags std::ios_base::boolalpha [static], [inherited]
```

Insert/extract `bool` in alphabetic rather than numeric format.
Referenced by std::boolalpha(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::noboolalpha().

cur

```
const seekdir std::ios_base::cur [static], [inherited]
```

Request a seek relative to the current position within the sequence.
Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

dec

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.
Referenced by std::dec().

end

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.
Referenced by std::basic_filebuf< _CharT, _Traits >::open(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff().

eofbit

```
const iostate std::ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.
Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >:: std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::ge std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< char_type, traits_type >::g std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< char_type, traits_type >::read(), std::basic_istream< _CharT, _Traits std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< char_type, traits_ty std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

failbit

```
const iostate std::ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::ge std::basic_istream< char_type, traits_type >::get(), std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::oper std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::s std::basic_ostream< _CharT, _Traits >::seekp(), and std::basic_ostream< _CharT, _Traits >::seekp().

fixed

```
const fmtflags std::ios_base::fixed  [static], [inherited]
```
Generate floating-point output in fixed-point notation.
Referenced by std::fixed(), and std::hexfloat().

floatfield

```
const fmtflags std::ios_base::floatfield  [static], [inherited]
```
A mask of scientific|fixed. Useful for the 2-arg form of `setf`.
Referenced by std::defaultfloat(), std::fixed(), std::hexfloat(), and std::scientific().

goodbit

```
const iostate std::ios_base::goodbit  [static], [inherited]
```
Indicates all is well.
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::d std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >: std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore() std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ostream< _CharT, _Traits > std::basic_ostream< char_type, traits_type >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >:: std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits std::basic_istream< char_type, traits_type >::seekg(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Trait and std::ws().

hex

```
const fmtflags std::ios_base::hex  [static], [inherited]
```
Converts integer input or generates integer output in hexadecimal base.
Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::hex().

in

```
const openmode std::ios_base::in  [static], [inherited]
```
Open for input. Default for `ifstream` and fstream.
Referenced by std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_stringbuf< _CharT, _Traits, _ std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_filebuf< _CharT, _Traits >::underflow(), std::basic_stringbuf< _CharT, _Traits, _ and std::basic_filebuf< _CharT, _Traits >::xsgetn().

internal

```
const fmtflags std::ios_base::internal  [static], [inherited]
```
Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.
Referenced by std::internal().

left

```
const fmtflags std::ios_base::left  [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)
Referenced by std::num_put< _CharT, _OutIter >::do_put(), and std::left().

oct

```
const fmtflags std::ios_base::oct  [static], [inherited]
```
Converts integer input or generates integer output in octal base.
Referenced by std::oct().

out

```
const openmode std::ios_base::out  [static], [inherited]
```
Open for output. Default for `ofstream` and fstream.
Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_stringbuf< _CharT, _Traits, std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

right

```
const fmtflags std::ios_base::right  [static], [inherited]
```
Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)
Referenced by std::right().

scientific

```
const fmtflags std::ios_base::scientific  [static], [inherited]
```
Generates floating-point output in scientific notation.
Referenced by std::hexfloat(), and std::scientific().

showbase

```
const fmtflags std::ios_base::showbase  [static], [inherited]
```
Generates a prefix indicating the numeric base of generated integer output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::noshowbase(), and std::showbase().

showpoint

```
const fmtflags std::ios_base::showpoint  [static], [inherited]
```
Generates a decimal-point character unconditionally in generated floating-point output.
Referenced by std::noshowpoint(), and std::showpoint().

showpos

```
const fmtflags std::ios_base::showpos  [static], [inherited]
```
Generates a + sign in non-negative generated numeric output.
Referenced by std::noshowpos(), and std::showpos().

skipws

```
const fmtflags std::ios_base::skipws  [static], [inherited]
```
Skips leading white space before certain input operations.
Referenced by std::noskipws(), std::__detail::operator>>(), and std::skipws().

trunc

`const openmode std::ios_base::trunc [static], [inherited]`
Truncate an existing stream when opening. Default for `ofstream`.

unitbuf

`const fmtflags std::ios_base::unitbuf [static], [inherited]`
Flushes output after each output operation.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::nounitbuf(), and std::unitbuf().

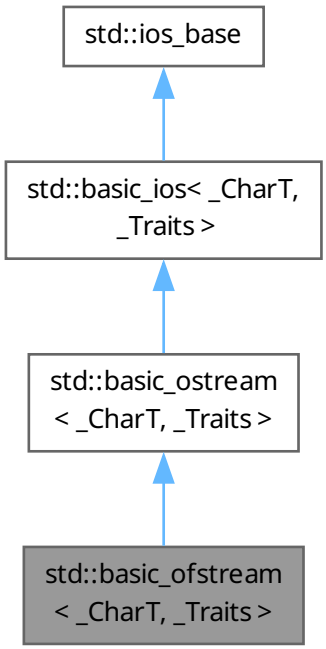
uppercase

`const fmtflags std::ios_base::uppercase [static], [inherited]`
Replaces certain lowercase letters with their uppercase equivalents in generated output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::nouppercase(), and std::uppercase().
The documentation for this class was generated from the following files:

- iosfwd
- sstream

5.227 std::basic_ofstream< _CharT, _Traits > Class Template Reference

`#include <fstream>`
Inheritance diagram for std::basic_ofstream< _CharT, _Traits >:



Public Types

- typedef ctype< _CharT > **__ctype_type**
- typedef basic_filebuf< char_type, traits_type > **__filebuf_type**
- typedef basic_ios< _CharT, _Traits > **__ios_type**
- typedef num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > **__num_put_type**
- typedef basic_ostream< char_type, traits_type > **__ostream_type**
- typedef basic_streambuf< _CharT, _Traits > **__streambuf_type**
- typedef _CharT **char_type**
- enum event { **erase_event** , **imbue_event** , **copyfmt_event** }
- typedef void(* event_callback) (event __e, ios_base &__b, int __i)
- typedef traits_type::int_type **int_type**
- typedef _Ios_Iostate iostate
- typedef traits_type::off_type **off_type**
- typedef _Ios_Openmode openmode
- typedef traits_type::pos_type **pos_type**
- typedef _Ios_Seekdir seekdir
- typedef _Traits **traits_type**

- typedef num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > __num_get_type

Public Member Functions

- basic_ofstream ()
- **basic_ofstream** (basic_ofstream &&__rhs)
- template<typename _Path, typename _Require = _If_fs_path<_Path>>
  basic_ofstream (const _Path &__s, ios_base::openmode __mode=ios_base::out)
- **basic_ofstream** (const basic_ofstream &)=delete
- basic_ofstream (const char *__s, ios_base::openmode __mode=ios_base::out)
- basic_ofstream (const std::string &__s, ios_base::openmode __mode=ios_base::out)
- ~basic_ofstream ()
- const locale & _M_getloc () const
- template<typename _ValueT>
  basic_ostream< _CharT, _Traits > & **_M_insert** (_ValueT __v)
- void **_M_setstate** (iostate __state)
- bool bad () const
- void clear (iostate __state=goodbit)
- void close ()
- basic_ios & copyfmt (const basic_ios &__rhs)
- bool eof () const
- iostate exceptions () const
- void exceptions (iostate __except)
- bool fail () const
- char_type fill () const
- char_type fill (char_type __ch)
- fmtflags flags () const
- fmtflags flags (fmtflags __fmtfl)
- __ostream_type & flush ()
- locale getloc () const
- bool good () const
- locale imbue (const locale &__loc)

- bool is_open ()
- bool **is_open** () const
- long & iword (int __ix)
- char narrow (char_type __c, char __dfault) const
- template<typename _Path>
  _If_fs_path< _Path, void > open (const _Path &__s, ios_base::openmode __mode=ios_base::out)
- void open (const char *__s, ios_base::openmode __mode=ios_base::out)
- void open (const std::string &__s, ios_base::openmode __mode=ios_base::out)
- __ostream_type & operator<< (__streambuf_type *__sb)
- __ostream_type & operator<< (const void *__p)
- __ostream_type & **operator**<< (nullptr_t)
- basic_ofstream & **operator=** (basic_ofstream &&__rhs)
- basic_ofstream & **operator=** (const basic_ofstream &)=delete
- streamsize precision () const
- streamsize precision (streamsize __prec)
- void *& pword (int __ix)
- basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > *__sb)
- __filebuf_type * rdbuf () const
- iostate rdstate () const
- void register_callback (event_callback __fn, int __index)
- __ostream_type & seekp (off_type, ios_base::seekdir)
- __ostream_type & seekp (pos_type)
- fmtflags setf (fmtflags __fmtfl)
- fmtflags setf (fmtflags __fmtfl, fmtflags __mask)
- void setstate (iostate __state)
- void **swap** (basic_ofstream &__rhs)
- pos_type tellp ()
- basic_ostream< _CharT, _Traits > * tie () const
- basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > *__tiestr)
- void unsetf (fmtflags __mask)
- char_type widen (char __c) const
- streamsize width () const
- streamsize width (streamsize __wide)

- __ostream_type & operator<< (__ostream_type &(*__pf)(__ostream_type &))
- __ostream_type & operator<< (__ios_type &(*__pf)(__ios_type &))
- __ostream_type & operator<< (ios_base &(*__pf)(ios_base &))

Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type std::basic_ostream::sentry. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, ios_base::badbit will be turned on in the stream's error state without causing an ios_base::failure to be thrown. The original exception will then be rethrown.

- __ostream_type & operator<< (long __n)
- __ostream_type & operator<< (unsigned long __n)
- __ostream_type & operator<< (bool __n)
- __ostream_type & operator<< (short __n)
- __ostream_type & operator<< (unsigned short __n)

- __ostream_type & operator<< (int __n)
- __ostream_type & operator<< (unsigned int __n)
- __ostream_type & operator<< (long long __n)
- __ostream_type & operator<< (unsigned long long __n)

- __ostream_type & operator<< (double __f)
- __ostream_type & operator<< (float __f)
- __ostream_type & operator<< (long double __f)

Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type std::basic_ostream::sentry. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, ios_base::badbit will be turned on in the stream's error state. If badbit is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- __ostream_type & put (char_type __c)
- __ostream_type & write (const char_type *__s, streamsize __n)

- operator bool () const
- bool operator! () const

Static Public Member Functions

- static bool sync_with_stdio (bool __sync=true)
- static int xalloc () throw ()

Public Attributes

- class __attribute((__abi_tag__("cxx11"))) failure typedef _Ios_Fmtflags fmtflags

Static Public Attributes

- static const openmode **__noreplace**
- static const fmtflags adjustfield
- static const openmode app
- static const openmode ate
- static const iostate badbit
- static const fmtflags basefield
- static const seekdir beg
- static const openmode binary
- static const fmtflags boolalpha
- static const seekdir cur
- static const fmtflags dec
- static const seekdir end
- static const iostate eofbit
- static const iostate failbit
- static const fmtflags fixed
- static const fmtflags floatfield
- static const iostate goodbit
- static const fmtflags hex

- static const openmode in
- static const fmtflags internal
- static const fmtflags left
- static const fmtflags oct
- static const openmode out
- static const fmtflags right
- static const fmtflags scientific
- static const fmtflags showbase
- static const fmtflags showpoint
- static const fmtflags showpos
- static const fmtflags skipws
- static const openmode trunc
- static const fmtflags unitbuf
- static const fmtflags uppercase

Protected Types

- enum { **_S_local_word_size** }

Protected Member Functions

- void **_M_cache_locale** (const locale &__loc)
- void **_M_call_callbacks** (event __ev) throw ()
- void **_M_dispose_callbacks** (void) throw ()
- _Words & **_M_grow_words** (int __index, bool __iword)
- void **_M_init** () throw ()
- template<typename _ValueT>
  __ostream_type & **_M_insert** (_ValueT __v)
- void **_M_move** (ios_base &) noexcept
- void **_M_swap** (ios_base &__rhs) noexcept
- void init (basic_streambuf< _CharT, _Traits > *__sb)
- void **move** (basic_ios &&__rhs)
- void **move** (basic_ios &__rhs)
- void **set_rdbuf** (basic_streambuf< _CharT, _Traits > *__sb)
- void **swap** (basic_ios &__rhs) noexcept
- void **swap** (basic_ostream &__rhs)

Protected Attributes

- _Callback_list * **_M_callbacks**
- const __ctype_type * **_M_ctype**
- iostate **_M_exception**
- char_type **_M_fill**
- bool **_M_fill_init**
- fmtflags **_M_flags**
- locale **_M_ios_locale**
- _Words **_M_local_word** [_S_local_word_size]
- const __num_get_type * **_M_num_get**
- const __num_put_type * **_M_num_put**
- streamsize **_M_precision**
- basic_streambuf< _CharT, _Traits > * **_M_streambuf**
- iostate **_M_streambuf_state**

- basic_ostream< _CharT, _Traits > * **_M_tie**
- streamsize **_M_width**
- _Words * **_M_word**
- int **_M_word_size**
- _Words **_M_word_zero**

5.227.1 Detailed Description

template<typename _CharT, typename _Traits>
class std::basic_ofstream< _CharT, _Traits >

Controlling output for files.

Template Parameters

_CharT	Type of character stream.
_Traits	Traits for character type, defaults to char_traits<_CharT>.

This class supports reading from named files, using the inherited functions from std::basic_ostream. To control the associated sequence, an instance of std::basic_filebuf is used, which this page refers to as `sb`.

5.227.2 Member Typedef Documentation

__num_get_type

```
template<typename _CharT, typename _Traits>
typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits> > std::basic_ios< _CharT, _Traits
>::__num_get_type  [inherited]
```

These are non-standard types.

event_callback

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i)  [inherited]
```

The type of an event callback function.

Parameters

_e	One of the members of the event enum.
_b	Reference to the ios_base object.
_i	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several ios_base and basic_ios functions, specifically imbue(), copyfmt(), and ~ios().

iostate

```
typedef _Ios_Iostate std::ios_base::iostate  [inherited]
```

This is a bitmask type.
_Ios_Iostate `is` implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit

- eofbit
- failbit
- goodbit

openmode

`typedef _Ios_Openmode std::ios_base::openmode [inherited]`
This is a bitmask type.
_Ios_Openmode is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type openmode are:

- app
- ate
- binary
- in
- out
- trunc

seekdir

`typedef _Ios_Seekdir std::ios_base::seekdir [inherited]`
This is an enumerated type.
_Ios_Seekdir is implementation-defined. Defined values of type seekdir are:

- beg
- cur, equivalent to SEEK_CUR in the C standard library.
- end, equivalent to SEEK_END in the C standard library.

5.227.3 Member Enumeration Documentation

event

`enum std::ios_base::event [inherited]`
The set of events that may be passed to an event callback.
erase_event is used during ~ios() and copyfmt(). imbue_event is used during imbue(). copyfmt_event is used during copyfmt().

5.227.4 Constructor & Destructor Documentation

basic_ofstream() [1/4]

```
template<typename _CharT, typename _Traits>
std::basic_ofstream< _CharT, _Traits >::basic_ofstream () [inline]
```
Default constructor.
Initializes sb using its default constructor, and passes &sb to the base class initializer. Does not open any files (you haven't given it a filename to open).

basic_ofstream() [2/4]

```
template<typename _CharT, typename _Traits>
std::basic_ofstream< _CharT, _Traits >::basic_ofstream (
            const char * __s,
            ios_base::openmode __mode = ios_base::out)  [inline], [explicit]
```

Create an output file stream.

Parameters

__s	Null terminated string specifying the filename.
__mode	Open file in specified mode (see std::ios_base).

`ios_base::out` is automatically included in *__mode*.

basic_ofstream() [3/4]

```
template<typename _CharT, typename _Traits>
std::basic_ofstream< _CharT, _Traits >::basic_ofstream (
            const std::string & __s,
            ios_base::openmode __mode = ios_base::out)  [inline], [explicit]
```

Create an output file stream.

Parameters

__s	std::string specifying the filename.
__mode	Open file in specified mode (see std::ios_base).

`ios_base::out` is automatically included in *__mode*.

basic_ofstream() [4/4]

```
template<typename _CharT, typename _Traits>
template<typename _Path, typename _Require = _If_fs_path<_Path>>
std::basic_ofstream< _CharT, _Traits >::basic_ofstream (
            const _Path & __s,
            ios_base::openmode __mode = ios_base::out)  [inline]
```

Create an output file stream.

Parameters

__s	filesystem::path specifying the filename.
__mode	Open file in specified mode (see std::ios_base).

`ios_base::out` is automatically included in *__mode*.

~basic_ofstream()

```
template<typename _CharT, typename _Traits>
std::basic_ofstream< _CharT, _Traits >::~basic_ofstream ()  [inline]
```

The destructor does nothing.
The file is closed by the filebuf object, not the formatting stream.

5.227.5 Member Function Documentation

_M_getloc()

```
const locale & std::ios_base::_M_getloc () const  [inline], [inherited]
```

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.
Referenced by std::money_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter
std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIte
std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::num_put< _CharT, _OutIter
std::time_put< _CharT, _OutIter >::do_put(), std::time_get< _CharT, _InIter >::get(), and std::time_put< _CharT, _OutIter >::put().

bad()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry().

clear()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::clear (
            iostate __state = goodbit)  [inherited]
```

[Re]sets the error state.

Parameters

__state	The new state flag(s) to set.

See std::ios_base::iostate for the possible bit values. Most users will not need to pass an argument.
References std::ios_base::badbit, exceptions(), rdbuf(), and rdstate().
Referenced by std::basic_ios< char_type, traits_type >::exceptions(), std::__detail::operator>>(), std::basic_istream< _CharT, _Traits >
rdbuf(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char_type, traits
and std::basic_istream< _CharT, _Traits >::unget().

close()

```
template<typename _CharT, typename _Traits>
void std::basic_ofstream< _CharT, _Traits >::close ()  [inline]
```

Close the file.
Calls `std::basic_filebuf::close()`. If that function fails, `failbit` is set in the stream's error state.

copyfmt()

```
template<typename _CharT, typename _Traits>
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
            const basic_ios< _CharT, _Traits > & __rhs)  [inherited]
```

Copies fields of __rhs into this.

Parameters

__rhs	The source values for the copies.

Returns

Reference to this object.

All fields of __rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase_event. After copying, each (new) callback is invoked with copyfmt_event. The final step is to copy exceptions().
References basic_ios(), std::__addressof(), exceptions(), fill(), std::ios_base::flags(), std::ios_base::getloc(), std::ios_base::precision(), tie(), std::tie(), and std::ios_base::width().

eof()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

exceptions() [1/2]

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions () const  [inline], [nodiscard], [inherited]
```

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.
Referenced by clear(), and copyfmt().

exceptions() [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
            iostate __except)  [inline], [inherited]
```

Throwing exceptions on errors.

Parameters

__except	The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type std::ios_base::failure is thrown.
If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
```

```
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr « "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr « "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.
Referenced by std::basic_ios< char_type, traits_type >::operator bool(), std::basic_ios< char_type, traits_type >::operator!(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >
std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::
and std::regex_traits< _Ch_type >::value().

fill() [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const  [inline], [nodiscard], [inherited]
```

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.
Referenced by copyfmt().

fill() [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
             char_type __ch)  [inline], [inherited]
```

Sets a new *empty* character.

Parameters

__ch	The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

flags() [1/2]

```
fmtflags std::ios_base::flags () const  [inline], [nodiscard], [inherited]
```
Access to format flags.

Returns

The format control flags for both input and output.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >
std::num_put< _CharT, _OutIter >::do_put(), std::num_put< _CharT, _OutIter >::do_put(), std::chrono::operator<<(),
std::operator<<(), std::__detail::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(),
std::operator>>(), std::operator>>(), and std::operator>>().

flags() [2/2]

```
fmtflags std::ios_base::flags (
            fmtflags __fmtfl)  [inline], [inherited]
```
Setting new format flags all at once.

Parameters

__fmtfl	The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *__fmtfl*.
References fmtflags.

flush()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::flush ()  [inherited]
```
Synchronizing the stream buffer.

Returns

*this

If `rdbuf()` is a null pointer, changes nothing.
Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit.
References std::ios_base::badbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits

getloc()

```
locale std::ios_base::getloc () const  [inline], [nodiscard], [inherited]
```
Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std↩
::locale()`, the global C++ locale.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::money_put< _CharT, _OutIter >::do_put(), std::basic_ios< _CharT, _Tr
std::chrono::operator<<(), std::operator>>(), std::operator>>(), and std::ws().

good()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if no error flags are set.

A wrapper around rdstate.
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::__detail::operator>>().

imbue()

```
template<typename _CharT, typename _Traits>
locale std::basic_ios< _CharT, _Traits >::imbue (
            const locale & __loc)  [inherited]
```

Moves to a new locale.

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.
Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html
References std::ios_base::getloc(), std::ios_base::imbue(), and rdbuf().
Referenced by std::chrono::operator<<().

init()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::init (
            basic_streambuf< _CharT, _Traits > * __sb)  [protected], [inherited]
```

All setup is performed here.
This is called from the public constructor. It is not virtual and cannot be redefined.
Referenced by std::basic_ios< char_type, traits_type >::basic_ios(), and std::basic_ostream< char_type, traits_type >::basic_ostream().

is_open()

```
template<typename _CharT, typename _Traits>
bool std::basic_ofstream< _CharT, _Traits >::is_open ()  [inline], [nodiscard]
```

Wrapper to test for an open file.

Returns

`rdbuf()->is_open()`

iword()

```
long & std::ios_base::iword (
            int __ix)  [inline], [inherited]
```

Access to integer array.

Parameters

_↩ _ix	Index into the array.

Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.
References iword().
Referenced by iword().

narrow()

```
template<typename _CharT, typename _Traits>
char std::basic_ios< _CharT, _Traits >::narrow (
            char_type __c,
            char __dfault) const  [inline], [inherited]
```

Squeezes characters.

Parameters

__c	The character to narrow.
__dfault	The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.
Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c,dfault)
```

Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.↩ html

open() [1/3]

```
template<typename _CharT, typename _Traits>
template<typename _Path>
_If_fs_path< _Path, void > std::basic_ofstream< _CharT, _Traits >::open (
            const _Path & __s,
            ios_base::openmode __mode = ios_base::out)  [inline]
```

Opens an external file.

Parameters

__s	The name of the file, as a filesystem::path.
__mode	The open mode flags.

Calls `std::basic_filebuf::open(__s,__mode|out)`. If that function fails, `failbit` is set in the stream's error state.

open() [2/3]

```
template<typename _CharT, typename _Traits>
void std::basic_ofstream< _CharT, _Traits >::open (
            const char * __s,
            ios_base::openmode __mode = ios_base::out)  [inline]
```

Opens an external file.

Parameters

__s	The name of the file.
__mode	The open mode flags.

Calls `std::basic_filebuf::open`(__s,__mode|out). If that function fails, `failbit` is set in the stream's error state.

open() [3/3]

```
template<typename _CharT, typename _Traits>
void std::basic_ofstream< _CharT, _Traits >::open (
            const std::string & __s,
            ios_base::openmode __mode = ios_base::out)  [inline]
```

Opens an external file.

Parameters

__s	The name of the file.
__mode	The open mode flags.

Calls `std::basic_filebuf::open`(s,mode|out). If that function fails, `failbit` is set in the stream's error state.

operator bool()

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool () const  [inline], [explicit], [nodiscard],
[inherited]
```

The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

operator"!()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator!  () const  [inline], [nodiscard], [inherited]
```

The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

operator<<() [1/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            __ios_type &(* __pf )(__ios_type &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

operator<<() [2/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            __ostream_type &(* __pf )(__ostream_type &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

operator<<() [3/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
            __streambuf_type * __sb)  [inherited]
```

Extracting from another streambuf.

Parameters

__sb	A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.
If `__sb` is NULL, the stream will set failbit in its error state.
Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets failbit in the error state

If the function inserts no characters, failbit is set.

operator<<() [4/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            bool __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

__↩ n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [5/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            const void * __p)  [inline], [inherited]
```

Pointer arithmetic inserters.

Parameters

_↩ _*p*	A variable of pointer type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [6/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            double __f)  [inline], [inherited]
```

Floating point arithmetic inserters.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [7/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            float __f)  [inline], [inherited]
```

Floating point arithmetic inserters.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [8/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
            int __n)  [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _*n*	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [9/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            ios_base &(* __pf )(ios_base &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

operator<<() [10/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            long __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _*n*	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [11/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            long double __f)  [inline], [inherited]
```

Floating point arithmetic inserters.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [12/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            long long __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [13/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
            short __n)  [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.
References basic_ostream(), std::ios_base::badbit, std::ostreambuf_iterator< _CharT, _Traits >::failed(), std::ios_base::goodbit, std::num_put< _CharT, _OutIter >::put(), std::basic_ios< _CharT, _Traits >::setstate(), and std::use_facet().

operator<<() [14/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned int __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [15/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned long __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [16/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned long long __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [17/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned short __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

precision() [1/2]

```
streamsize std::ios_base::precision () const  [inline], [nodiscard], [inherited]
```

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::chrono::operator<<().

precision() [2/2]

```
streamsize std::ios_base::precision (
            streamsize __prec)  [inline], [inherited]
```

Changing flags.

Parameters

__prec	The new precision value.

Returns

The previous value of precision().

put()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (
            char_type __c)  [inherited]
```

Simple insertion.

Parameters

__c	The character to insert.

Returns

*this

Tries to insert __c.

Note

This function is not overloaded on signed char and unsigned char.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits

pword()

```
void *& std::ios_base::pword (
            int __ix)  [inline], [inherited]
```

Access to void pointer array.

Parameters

_↩ _*ix*	Index into the array.

Returns

A reference to a void* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

rdbuf() [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
            basic_streambuf< _CharT, _Traits > * __sb)  [inherited]
```

Changing the underlying buffer.

Parameters

__sb	The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.
Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream     foo;         // or some other derived type
std::streambuf*  p = .....;

foo.ios::rdbuf(p);            // ios == basic_ios<char>
```

References clear().

rdbuf() [2/2]

```
template<typename _CharT, typename _Traits>
__filebuf_type * std::basic_ofstream< _CharT, _Traits >::rdbuf () const  [inline], [nodiscard]
```

Accessing the underlying buffer.

Returns

The current basic_filebuf buffer.

This hides both signatures of std::basic_ios::rdbuf().

rdstate()

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::rdstate () const  [inline], [nodiscard], [inherited]
```

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See std::ios_base::iostate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().
Referenced by std::basic_ios< char_type, traits_type >::bad(), clear(), std::basic_ios< char_type, traits_type >::eof(), std::basic_ios< char_type, traits_type >::fail(), std::basic_ios< char_type, traits_type >::good(), std::basic_istream< _CharT, _Traits >::
std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char_type, traits_type >
and std::basic_istream< _CharT, _Traits >::unget().

register_callback()

```
void std::ios_base::register_callback (
            event_callback __fn,
            int __index)  [inherited]
```

Add the callback __fn with parameter __index.

Parameters

__fn	The function to add.
__index	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.
References fmtflags.

seekp() [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
            off_type __off,
            ios_base::seekdir __dir)  [inherited]
```

Changing the current write position.

Parameters

__off	A file offset object.
__dir	The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off,dir)`. If that function fails, sets failbit.
References std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::out, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

seekp() [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
            pos_type __pos)  [inherited]
```

Changing the current write position.

Parameters

__pos	A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.
References std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::out, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

setf() [1/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl)  [inline], [inherited]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.
References fmtflags.
Referenced by std::__detail::operator>>().

setf() [2/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl,
            fmtflags __mask)  [inline], [inherited]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.
__mask	The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.
References fmtflags.

setstate()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
            iostate __state)  [inline], [inherited]
```

Sets additional flags in the error state.

Parameters

__state	The additional state flag(s) to set.

See std::ios_base::iostate for the possible bit values.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::getline(), std::getline(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _Cha std::operator>>(), std::operator>>(), std::operator>>(), std::tr2::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >: std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits > std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), and std::ws().

sync_with_stdio()

```
static bool std::ios_base::sync_with_stdio (
            bool __sync = true)  [static], [inherited]
```

Interaction with the standard C I/O objects.

Parameters

__sync	Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary

tellp()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits >::pos_type std::basic_ostream< _CharT, _Traits >::tellp ()  [inherited]
```

Getting the current write position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type`(-1) to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0,cur,out)`.
References std::ios_base::cur, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::out, and std::basic_ios< _CharT, _Traits >::rdbuf()

tie() [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie () const  [inline],
[nodiscard], [inherited]
```

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and copyfmt().

tie() [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie (
            basic_ostream< _CharT, _Traits > * __tiestr)  [inline], [inherited]
```

Ties this stream to an output stream.

Parameters

__tiestr	The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

unsetf()

```
void std::ios_base::unsetf (
            fmtflags __mask)  [inline], [inherited]
```

Clearing format flags.

Parameters

__mask	The flags to unset.

This function clears *__mask* in the format flags.
References fmtflags.

widen()

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
            char __c) const  [inline], [inherited]
```

Widens characters.

Parameters

__c	The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.
Returns the result of
`std::use_facet<ctype<char_type> >(getloc()).widen(c)`
Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html
Referenced by std::basic_ios< char_type, traits_type >::fill(), std::getline(), std::getline(), and std::tr2::operator>>().

width() [1/2]

```
streamsize std::ios_base::width () const  [inline], [nodiscard], [inherited]
```

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_put< _CharT, _OutIter >::do_put(), std::operator>>(), and std::operator>>().

width() [2/2]

```
streamsize std::ios_base::width (
             streamsize __wide)  [inline], [inherited]
```

Changing flags.

Parameters

__wide	The new width value.

Returns

The previous value of width().

write()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::write (
             const char_type * __s,
             streamsize __n)  [inherited]
```

Character string insertion.

Parameters

_↩ _s	The array to insert.
_↩ _n	Maximum number of characters to insert.

Returns

*this

Characters are copied from __s and inserted into the stream until one of the following happens:

- __n characters are inserted
- inserting into the output sequence fails (in this case, badbit will be set in the stream's error state)

Note

This function is not overloaded on signed char and unsigned char.

xalloc()

```
static int std::ios_base::xalloc () throw ( )   [static], [inherited]
```

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the iword and pword functions. The expectation is that an application calls xalloc in order to obtain an index in the iword and pword arrays that can be used without fear of conflict.
The implementation maintains a static variable that is incremented and returned on each invocation. xalloc is guaranteed to return an index that is safe to use in the iword and pword arrays.

5.227.6 Member Data Documentation

adjustfield

```
const fmtflags std::ios_base::adjustfield  [static], [inherited]
```

A mask of left|right|internal. Useful for the 2-arg form of `setf`.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::internal(), std::left(), and std::right().

app

```
const openmode std::ios_base::app  [static], [inherited]
```

Seek to end before each write.
Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

ate

```
const openmode std::ios_base::ate  [static], [inherited]
```

Open and seek to end immediately after opening.
Referenced by std::basic_filebuf< _CharT, _Traits >::open().

badbit

```
const iostate std::ios_base::badbit  [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< char_type, traits_type >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::basic_ios< _CharT, _Traits >::clear(), std::basic_ostream< _CharT, _Trai std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_ostream< _CharT, _Traits >::opera std::basic_ostream< char_type, traits_type >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >: std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< char_type, traits_ty std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_istream< char_type, traits_type and std::ws().

basefield

```
const fmtflags std::ios_base::basefield  [static], [inherited]
```

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.
Referenced by std::dec(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::hex(), and std::oct().

beg

const seekdir std::ios_base::beg [static], [inherited]
Request a seek relative to the beginning of the stream.
Referenced by std::basic_filebuf< _CharT, _Traits >::seekpos().

binary

const openmode std::ios_base::binary [static], [inherited]
Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary.
Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

boolalpha

const fmtflags std::ios_base::boolalpha [static], [inherited]
Insert/extract bool in alphabetic rather than numeric format.
Referenced by std::boolalpha(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::noboolalpha().

cur

const seekdir std::ios_base::cur [static], [inherited]
Request a seek relative to the current position within the sequence.
Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

dec

const fmtflags std::ios_base::dec [static], [inherited]
Converts integer input or generates integer output in decimal base.
Referenced by std::dec().

end

const seekdir std::ios_base::end [static], [inherited]
Request a seek relative to the current end of the sequence.
Referenced by std::basic_filebuf< _CharT, _Traits >::open(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff().

eofbit

const iostate std::ios_base::eofbit [static], [inherited]
Indicates that an input operation reached the end of an input sequence.
Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >:
std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(),
std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter
std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::ge
std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< char_type, traits_type >::g
std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::putback(),
std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< char_type, traits_type >::read(), std::basic_istream< _CharT, _Traits
std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< char_type, traits_ty
std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

failbit

```
const iostate std::ios_base::failbit  [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::ge std::basic_istream< char_type, traits_type >::get(), std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::oper std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::s std::basic_ostream< _CharT, _Traits >::seekp(), and std::basic_ostream< _CharT, _Traits >::seekp().

fixed

```
const fmtflags std::ios_base::fixed  [static], [inherited]
```

Generate floating-point output in fixed-point notation.

Referenced by std::fixed(), and std::hexfloat().

floatfield

```
const fmtflags std::ios_base::floatfield  [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Referenced by std::defaultfloat(), std::fixed(), std::hexfloat(), and std::scientific().

goodbit

```
const iostate std::ios_base::goodbit  [static], [inherited]
```

Indicates all is well.

Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::d std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >: std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore() std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ostream< _CharT, _Traits > std::basic_ostream< char_type, traits_type >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >:: std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits std::basic_istream< char_type, traits_type >::seekg(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Trait and std::ws().

hex

```
const fmtflags std::ios_base::hex  [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::hex().

in

```
const openmode std::ios_base::in  [static], [inherited]
```

Open for input. Default for `ifstream` and fstream.

Referenced by std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_stringbuf< _CharT, _Traits, _

std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_filebuf< _CharT, _Traits >::underflow(), std::basic_stringbuf< _CharT, _Traits, _ and std::basic_filebuf< _CharT, _Traits >::xsgetn().

internal

```
const fmtflags std::ios_base::internal  [static], [inherited]
```

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.
Referenced by std::internal().

left

```
const fmtflags std::ios_base::left  [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)
Referenced by std::num_put< _CharT, _OutIter >::do_put(), and std::left().

oct

```
const fmtflags std::ios_base::oct  [static], [inherited]
```

Converts integer input or generates integer output in octal base.
Referenced by std::oct().

out

```
const openmode std::ios_base::out  [static], [inherited]
```

Open for output. Default for `ofstream` and fstream.
Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_stringbuf< _CharT, _Traits, std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

right

```
const fmtflags std::ios_base::right  [static], [inherited]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)
Referenced by std::right().

scientific

```
const fmtflags std::ios_base::scientific  [static], [inherited]
```

Generates floating-point output in scientific notation.
Referenced by std::hexfloat(), and std::scientific().

showbase

```
const fmtflags std::ios_base::showbase  [static], [inherited]
```

Generates a prefix indicating the numeric base of generated integer output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::noshowbase(), and std::showbase().

showpoint

```
const fmtflags std::ios_base::showpoint  [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.
Referenced by std::noshowpoint(), and std::showpoint().

showpos

`const fmtflags std::ios_base::showpos [static], [inherited]`
Generates a + sign in non-negative generated numeric output.
Referenced by std::noshowpos(), and std::showpos().

skipws

`const fmtflags std::ios_base::skipws [static], [inherited]`
Skips leading white space before certain input operations.
Referenced by std::noskipws(), std::__detail::operator>>(), and std::skipws().

trunc

`const openmode std::ios_base::trunc [static], [inherited]`
Truncate an existing stream when opening. Default for `ofstream`.

unitbuf

`const fmtflags std::ios_base::unitbuf [static], [inherited]`
Flushes output after each output operation.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::nounitbuf(), and std::unitbuf().

uppercase

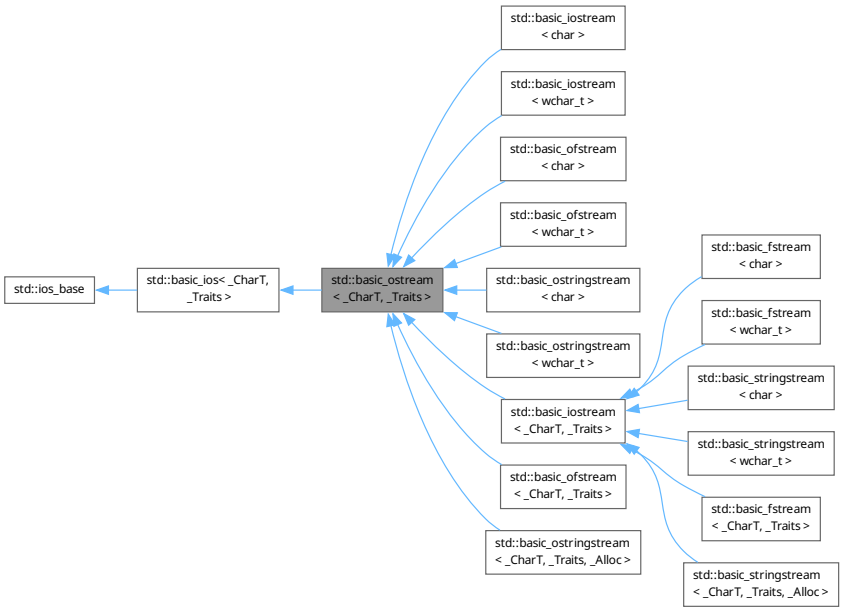
`const fmtflags std::ios_base::uppercase [static], [inherited]`
Replaces certain lowercase letters with their uppercase equivalents in generated output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::nouppercase(), and std::uppercase().
The documentation for this class was generated from the following file:

- fstream

5.228 std::basic_ostream< _CharT, _Traits > Class Template Reference

`#include <ostream.h>`

Inheritance diagram for std::basic_ostream< _CharT, _Traits >:



Classes

- class sentry

Public Types

- typedef ctype< _CharT > **__ctype_type**
- typedef basic_ios< _CharT, _Traits > **__ios_type**
- typedef num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > **__num_put_type**
- typedef basic_ostream< _CharT, _Traits > **__ostream_type**
- typedef basic_streambuf< _CharT, _Traits > **__streambuf_type**
- typedef _CharT **char_type**
- enum event { **erase_event** , **imbue_event** , **copyfmt_event** }
- typedef void(* event_callback) (event __e, ios_base &__b, int __i)
- typedef _Traits::int_type **int_type**
- typedef _Ios_Iostate iostate
- typedef _Traits::off_type **off_type**
- typedef _Ios_Openmode openmode
- typedef _Traits::pos_type **pos_type**
- typedef _Ios_Seekdir seekdir
- typedef _Traits **traits_type**

- typedef num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > __num_get_type

Public Member Functions

- basic_ostream (__streambuf_type *__sb)
- virtual ~basic_ostream ()
- const locale & _M_getloc () const
- template<typename _ValueT>
  basic_ostream< _CharT, _Traits > & **_M_insert** (_ValueT __v)
- void **_M_setstate** (iostate __state)
- bool bad () const
- void clear (iostate __state=goodbit)
- basic_ios & copyfmt (const basic_ios &__rhs)
- bool eof () const
- iostate exceptions () const
- void exceptions (iostate __except)
- bool fail () const
- char_type fill () const
- char_type fill (char_type __ch)
- fmtflags flags () const
- fmtflags flags (fmtflags __fmtfl)
- __ostream_type & flush ()
- locale getloc () const
- bool good () const
- locale imbue (const locale &__loc)
- long & iword (int __ix)
- char narrow (char_type __c, char __dfault) const
- __ostream_type & operator<< (__streambuf_type *__sb)
- __ostream_type & operator<< (const void *__p)
- __ostream_type & **operator**<< (nullptr_t)
- streamsize precision () const
- streamsize precision (streamsize __prec)
- void *& pword (int __ix)
- basic_streambuf< _CharT, _Traits > * rdbuf () const
- basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > *__sb)
- iostate rdstate () const
- void register_callback (event_callback __fn, int __index)
- __ostream_type & seekp (off_type, ios_base::seekdir)
- __ostream_type & seekp (pos_type)
- fmtflags setf (fmtflags __fmtfl)
- fmtflags setf (fmtflags __fmtfl, fmtflags __mask)
- void setstate (iostate __state)
- pos_type tellp ()
- basic_ostream< _CharT, _Traits > * tie () const
- basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > *__tiestr)
- void unsetf (fmtflags __mask)
- char_type widen (char __c) const
- streamsize width () const
- streamsize width (streamsize __wide)

- __ostream_type & operator<< (__ostream_type &(*__pf)(__ostream_type &))
- __ostream_type & operator<< (__ios_type &(*__pf)(__ios_type &))
- __ostream_type & operator<< (ios_base &(*__pf)(ios_base &))

Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type std::basic_ostream::sentry. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, ios_base::badbit will be turned on in the stream's error state without causing an ios_base::failure to be thrown. The original exception will then be rethrown.

- __ostream_type & operator<< (long __n)
- __ostream_type & operator<< (unsigned long __n)
- __ostream_type & operator<< (bool __n)
- __ostream_type & operator<< (short __n)
- __ostream_type & operator<< (unsigned short __n)
- __ostream_type & operator<< (int __n)
- __ostream_type & operator<< (unsigned int __n)
- __ostream_type & operator<< (long long __n)
- __ostream_type & operator<< (unsigned long long __n)

- __ostream_type & operator<< (double __f)
- __ostream_type & operator<< (float __f)
- __ostream_type & operator<< (long double __f)

Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type std::basic_ostream::sentry. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, ios_base::badbit will be turned on in the stream's error state. If badbit is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- __ostream_type & put (char_type __c)
- __ostream_type & write (const char_type *__s, streamsize __n)

- operator bool () const
- bool operator! () const

Static Public Member Functions

- static bool sync_with_stdio (bool __sync=true)
- static int xalloc () throw ()

Public Attributes

- class __attribute((__abi_tag__("cxx11"))) failure typedef _Ios_Fmtflags fmtflags

Static Public Attributes

- static const openmode **__noreplace**
- static const fmtflags adjustfield
- static const openmode app
- static const openmode ate
- static const iostate badbit
- static const fmtflags basefield

- static const seekdir beg
- static const openmode binary
- static const fmtflags boolalpha
- static const seekdir cur
- static const fmtflags dec
- static const seekdir end
- static const iostate eofbit
- static const iostate failbit
- static const fmtflags fixed
- static const fmtflags floatfield
- static const iostate goodbit
- static const fmtflags hex
- static const openmode in
- static const fmtflags internal
- static const fmtflags left
- static const fmtflags oct
- static const openmode out
- static const fmtflags right
- static const fmtflags scientific
- static const fmtflags showbase
- static const fmtflags showpoint
- static const fmtflags showpos
- static const fmtflags skipws
- static const openmode trunc
- static const fmtflags unitbuf
- static const fmtflags uppercase

Protected Types

- enum { **_S_local_word_size** }

Protected Member Functions

- **basic_ostream** (basic_iostream< _CharT, _Traits > &)
- **basic_ostream** (basic_ostream &&__rhs)
- **basic_ostream** (const basic_ostream &)=delete
- void **_M_cache_locale** (const locale &__loc)
- void **_M_call_callbacks** (event __ev) throw ()
- void **_M_dispose_callbacks** (void) throw ()
- _Words & **_M_grow_words** (int __index, bool __iword)
- void **_M_init** () throw ()
- template<typename _ValueT>
  __ostream_type & **_M_insert** (_ValueT __v)
- void **_M_move** (ios_base &) noexcept
- void **_M_swap** (ios_base &__rhs) noexcept
- void init (basic_streambuf< _CharT, _Traits > *__sb)
- void **move** (basic_ios &&__rhs)
- void **move** (basic_ios &__rhs)
- basic_ostream & **operator=** (basic_ostream &&__rhs)
- basic_ostream & **operator=** (const basic_ostream &)=delete
- void **set_rdbuf** (basic_streambuf< _CharT, _Traits > *__sb)
- void **swap** (basic_ios &__rhs) noexcept
- void **swap** (basic_ostream &__rhs)

Protected Attributes

- _Callback_list * **_M_callbacks**
- const __ctype_type * **_M_ctype**
- iostate **_M_exception**
- char_type **_M_fill**
- bool **_M_fill_init**
- fmtflags **_M_flags**
- locale **_M_ios_locale**
- _Words **_M_local_word** [_S_local_word_size]
- const __num_get_type * **_M_num_get**
- const __num_put_type * **_M_num_put**
- streamsize **_M_precision**
- basic_streambuf< _CharT, _Traits > * **_M_streambuf**
- iostate **_M_streambuf_state**
- basic_ostream< _CharT, _Traits > * **_M_tie**
- streamsize **_M_width**
- _Words * **_M_word**
- int **_M_word_size**
- _Words **_M_word_zero**

Friends

- class **sentry**

5.228.1 Detailed Description

template<typename _CharT, typename _Traits>
class std::basic_ostream< _CharT, _Traits >

Template class basic_ostream.

Template Parameters

_CharT	Type of character stream.
_Traits	Traits for character type, defaults to char_traits<_CharT>.

This is the base class for all output streams. It provides text formatting of all builtin types, and communicates with any class derived from basic_streambuf to do the actual output.

5.228.2 Member Typedef Documentation

__num_get_type

```
template<typename _CharT, typename _Traits>
typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits> > std::basic_ios< _CharT, _Traits
>::__num_get_type  [inherited]
```

These are non-standard types.

event_callback

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i)  [inherited]
```

The type of an event callback function.

Parameters

_↩ _e	One of the members of the event enum.
_↩ _b	Reference to the ios_base object.
_↩ _i	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several ios_base and basic_ios functions, specifically imbue(), copyfmt(), and ~ios().

iostate

`typedef _Ios_Iostate std::ios_base::iostate [inherited]`
This is a bitmask type.
_Ios_Iostate is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit
- eofbit
- failbit
- goodbit

openmode

`typedef _Ios_Openmode std::ios_base::openmode [inherited]`
This is a bitmask type.
_Ios_Openmode is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type openmode are:

- app
- ate
- binary
- in
- out
- trunc

seekdir

`typedef _Ios_Seekdir std::ios_base::seekdir [inherited]`
This is an enumerated type.
_Ios_Seekdir is implementation-defined. Defined values of type seekdir are:

- beg
- cur, equivalent to SEEK_CUR in the C standard library.
- end, equivalent to SEEK_END in the C standard library.

5.228.3 Member Enumeration Documentation

event

```
enum std::ios_base::event  [inherited]
```

The set of events that may be passed to an event callback.
erase_event is used during ~ios() and copyfmt(). imbue_event is used during imbue(). copyfmt_event is used during copyfmt().

5.228.4 Constructor & Destructor Documentation

basic_ostream()

```
template<typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits >::basic_ostream (
            __streambuf_type * __sb)  [inline], [explicit]
```

Base constructor.
This ctor is almost never called by the user directly, rather from derived classes' initialization lists, which pass a pointer to their own stream buffer.
Referenced by operator<<().

~basic_ostream()

```
template<typename _CharT, typename _Traits>
virtual std::basic_ostream< _CharT, _Traits >::~basic_ostream ()  [inline], [virtual]
```

Base destructor.
This does very little apart from providing a virtual base dtor.

5.228.5 Member Function Documentation

_M_getloc()

```
const locale & std::ios_base::_M_getloc () const  [inline], [inherited]
```

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.
Referenced by std::money_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter
std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIte
std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::num_put< _CharT, _OutIter
std::time_put< _CharT, _OutIter >::do_put(), std::time_get< _CharT, _InIter >::get(), and std::time_put< _CharT, _OutIter >::put().

bad()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry().

clear()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::clear (
            iostate __state = goodbit)  [inherited]
```

[Re]sets the error state.

Parameters

__state	The new state flag(s) to set.

See std::ios_base::iostate for the possible bit values. Most users will not need to pass an argument.
References std::ios_base::badbit, exceptions(), rdbuf(), and rdstate().
Referenced by std::basic_ios< char_type, traits_type >::exceptions(), std::__detail::operator>>(), std::basic_istream< _CharT, _Traits >
rdbuf(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char_type, traits
and std::basic_istream< _CharT, _Traits >::unget().

copyfmt()

```
template<typename _CharT, typename _Traits>
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
            const basic_ios< _CharT, _Traits > & __rhs)  [inherited]
```

Copies fields of __rhs into this.

Parameters

__rhs	The source values for the copies.

Returns

Reference to this object.

All fields of __rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase_event. After copying, each (new) callback is invoked with copyfmt_event. The final step is to copy exceptions().
References basic_ios(), std::__addressof(), exceptions(), fill(), std::ios_base::flags(), std::ios_base::getloc(), std::ios_base::precision(), tie(), std::tie(), and std::ios_base::width().

eof()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

exceptions() [1/2]

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions () const  [inline], [nodiscard], [inherited]
```

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.
Referenced by clear(), and copyfmt().

exceptions() [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
            iostate __except)  [inline], [inherited]
```

Throwing exceptions on errors.

Parameters

__except	The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type std::ios_base::failure is thrown.
If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr « "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr « "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.
Referenced by std::basic_ios< char_type, traits_type >::operator bool(), std::basic_ios< char_type, traits_type >::operator!(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >
std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::
and std::regex_traits< _Ch_type >::value().

fill() [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const  [inline], [nodiscard], [inherited]
```

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.
Referenced by copyfmt().

fill() [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
            char_type __ch)  [inline], [inherited]
```

Sets a new *empty* character.

Parameters

__ch	The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

flags() [1/2]

```
fmtflags std::ios_base::flags () const  [inline], [nodiscard], [inherited]
```

Access to format flags.

Returns

The format control flags for both input and output.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >
std::num_put< _CharT, _OutIter >::do_put(), std::num_put< _CharT, _OutIter >::do_put(), std::chrono::operator<<(),
std::operator<<(), std::__detail::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(),
std::operator>>(), std::operator>>(), and std::operator>>().

flags() [2/2]

```
fmtflags std::ios_base::flags (
            fmtflags __fmtfl)  [inline], [inherited]
```

Setting new format flags all at once.

Parameters

__fmtfl	The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *__fmtfl*.
References fmtflags.

flush()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::flush ()
```

Synchronizing the stream buffer.

Returns

*this

If `rdbuf()` is a null pointer, changes nothing.
Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit.
References std::ios_base::badbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits

getloc()

```
locale std::ios_base::getloc () const  [inline], [nodiscard], [inherited]
```

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::money_put< _CharT, _OutIter >::do_put(), std::basic_ios< _CharT, _Tr
std::chrono::operator<<(), std::operator>>(), std::operator>>(), and std::ws().

good()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if no error flags are set.

A wrapper around rdstate.
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::__detail::operator>>().

imbue()

```
template<typename _CharT, typename _Traits>
locale std::basic_ios< _CharT, _Traits >::imbue (
            const locale & __loc)  [inherited]
```

Moves to a new locale.

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.
Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html
References std::ios_base::getloc(), std::ios_base::imbue(), and rdbuf().
Referenced by std::chrono::operator<<().

init()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::init (
            basic_streambuf< _CharT, _Traits > * __sb)  [protected], [inherited]
```

All setup is performed here.
This is called from the public constructor. It is not virtual and cannot be redefined.
Referenced by std::basic_ios< char_type, traits_type >::basic_ios(), and std::basic_ostream< char_type, traits_type >::basic_ostream().

iword()

```
long & std::ios_base::iword (
            int __ix)  [inline], [inherited]
```

Access to integer array.

Parameters

__ix	Index into the array.

Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.
References iword().
Referenced by iword().

narrow()

```
template<typename _CharT, typename _Traits>
char std::basic_ios< _CharT, _Traits >::narrow (
            char_type __c,
            char __dfault) const  [inline], [inherited]
```

Squeezes characters.

Parameters

__c	The character to narrow.
__dfault	The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.
Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c,dfault)
```

Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html

operator bool()

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool () const  [inline], [explicit], [nodiscard],
[inherited]
```

The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

operator"!()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator!  () const  [inline], [nodiscard], [inherited]
```

The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

operator<<() [1/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            __ios_type &(* __pf )(__ios_type &))  [inline]
```

Interface for manipulators.
Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

operator<<() [2/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            __ostream_type &(* __pf )(__ostream_type &))  [inline]
```

Interface for manipulators.
Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

operator<<() [3/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
            __streambuf_type * __sb)
```

Extracting from another streambuf.

Parameters

__sb	A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If __sb is NULL, the stream will set failbit in its error state.
Characters are extracted from __sb and inserted into *this until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from __sb, which sets failbit in the error state

If the function inserts no characters, failbit is set.

operator<<() [4/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            bool __n)  [inline]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [5/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            const void * __p)  [inline]
```

Pointer arithmetic inserters.

Parameters

_↩ _p	A variable of pointer type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [6/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            double __f)  [inline]
```

Floating point arithmetic inserters.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [7/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            float __f)  [inline]
```

Floating point arithmetic inserters.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [8/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
            int __n)
```

Integer arithmetic inserters.

Parameters

_↩ *_n*	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [9/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            ios_base &(* __pf )(ios_base &))  [inline]
```

Interface for manipulators.
Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

operator<<() [10/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            long __n)  [inline]
```

Integer arithmetic inserters.

Parameters

_↩ _*n*	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [11/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            long double __f)  [inline]
```

Floating point arithmetic inserters.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [12/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            long long __n)  [inline]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [13/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
            short __n)
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.
References basic_ostream(), std::ios_base::badbit, std::ostreambuf_iterator< _CharT, _Traits >::failed(), std::ios_base::goodbit, std::num_put< _CharT, _OutIter >::put(), std::basic_ios< _CharT, _Traits >::setstate(), and std::use_facet().

operator<<() [14/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned int __n)  [inline]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [15/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned long __n)  [inline]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [16/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned long long __n)  [inline]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [17/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned short __n)  [inline]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

precision() [1/2]

```
streamsize std::ios_base::precision () const  [inline], [nodiscard], [inherited]
```

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::chrono::operator<<().

precision() [2/2]

```
streamsize std::ios_base::precision (
            streamsize __prec)  [inline], [inherited]
```

Changing flags.

Parameters

__prec	The new precision value.

Returns

The previous value of precision().

put()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (
            char_type __c)
```

Simple insertion.

Parameters

_↩ _c	The character to insert.

Returns

*this

Tries to insert __c.

Note

This function is not overloaded on signed char and unsigned char.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits

pword()

```
void *& std::ios_base::pword (
            int __ix)  [inline], [inherited]
```

Access to void pointer array.

Parameters

_↩ _ix	Index into the array.

Returns

A reference to a void* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

rdbuf() [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf () const  [inline],
[nodiscard], [inherited]
```

Accessing the underlying buffer.

Returns

The current stream buffer.

This does not change the state of the stream.
Referenced by clear(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getlin std::getline(), std::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::ignore(), imbue(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::tr2::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >: std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >: std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::t std::basic_ostream< _CharT, _Traits >::tellp(), and std::ws().

rdbuf() [2/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
            basic_streambuf< _CharT, _Traits > * __sb)  [inherited]
```

Changing the underlying buffer.

Parameters

__sb	The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.
Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream     foo;         // or some other derived type
std::streambuf*  p = .....;

foo.ios::rdbuf(p);            // ios == basic_ios<char>
```

References clear().

rdstate()

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::rdstate () const  [inline], [nodiscard], [inherited]
```

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See std::ios_base::iostate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().
Referenced by std::basic_ios< char_type, traits_type >::bad(), clear(), std::basic_ios< char_type, traits_type >::eof(), std::basic_ios< char_type, traits_type >::fail(), std::basic_ios< char_type, traits_type >::good(), std::basic_istream< _CharT, _Traits >::
std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char_type, traits_type >
and std::basic_istream< _CharT, _Traits >::unget().

register_callback()

```
void std::ios_base::register_callback (
            event_callback __fn,
            int __index)  [inherited]
```

Add the callback __fn with parameter __index.

Parameters

__fn	The function to add.
__index	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.
References fmtflags.

seekp() [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
            off_type __off,
            ios_base::seekdir __dir)
```

Changing the current write position.

Parameters

__off	A file offset object.
__dir	The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off,dir)`. If that function fails, sets failbit.
References std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::out, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

seekp() [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
            pos_type __pos)
```

Changing the current write position.

Parameters

__pos	A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.
References std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::out, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

setf() [1/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl)  [inline], [inherited]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.
References fmtflags.
Referenced by std::__detail::operator>>().

setf() [2/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl,
            fmtflags __mask)  [inline], [inherited]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.
__mask	The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.
References fmtflags.

setstate()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
            iostate __state)  [inline], [inherited]
```

Sets additional flags in the error state.

Parameters

__state	The additional state flag(s) to set.

See std::ios_base::iostate for the possible bit values.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::getline(), std::getline(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _Cha std::operator>>(), std::operator>>(), std::operator>>(), std::tr2::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >: std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits > std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), and std::ws().

sync_with_stdio()

```
static bool std::ios_base::sync_with_stdio (
            bool __sync = true)  [static], [inherited]
```

Interaction with the standard C I/O objects.

Parameters

__sync	Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary

tellp()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits >::pos_type std::basic_ostream< _CharT, _Traits >::tellp ()
```

Getting the current write position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type`(-1) to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0,cur,out)`.
References std::ios_base::cur, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::out, and std::basic_ios< _CharT, _Traits >::rdbuf()

tie() [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie () const  [inline],
[nodiscard], [inherited]
```

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and copyfmt().

tie() [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie (
            basic_ostream< _CharT, _Traits > * __tiestr)  [inline], [inherited]
```

Ties this stream to an output stream.

Parameters

__tiestr	The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

unsetf()

```
void std::ios_base::unsetf (
            fmtflags __mask)  [inline], [inherited]
```

Clearing format flags.

Parameters

__mask	The flags to unset.

This function clears *__mask* in the format flags.
References fmtflags.

widen()

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
            char __c) const  [inline], [inherited]
```

Widens characters.

Parameters

_c	The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.
Returns the result of
`std::use_facet<ctype<char_type> >(getloc()).widen(c)`
Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html
Referenced by std::basic_ios< char_type, traits_type >::fill(), std::getline(), std::getline(), and std::tr2::operator>>().

width() [1/2]

```
streamsize std::ios_base::width () const  [inline], [nodiscard], [inherited]
```

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_put< _CharT, _OutIter >::do_put(), std::operator>>(), and std::operator>>().

width() [2/2]

```
streamsize std::ios_base::width (
             streamsize __wide)  [inline], [inherited]
```

Changing flags.

Parameters

__wide	The new width value.

Returns

The previous value of width().

write()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::write (
             const char_type * __s,
             streamsize __n)
```

Character string insertion.

Parameters

_↩ _s	The array to insert.
_↩ _n	Maximum number of characters to insert.

Returns

*this

Characters are copied from __s and inserted into the stream until one of the following happens:

- __n characters are inserted
- inserting into the output sequence fails (in this case, badbit will be set in the stream's error state)

Note

This function is not overloaded on signed char and unsigned char.

xalloc()

```
static int std::ios_base::xalloc () throw ( )   [static], [inherited]
```

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the iword and pword functions. The expectation is that an application calls xalloc in order to obtain an index in the iword and pword arrays that can be used without fear of conflict.
The implementation maintains a static variable that is incremented and returned on each invocation. xalloc is guaranteed to return an index that is safe to use in the iword and pword arrays.

5.228.6 Member Data Documentation

adjustfield

```
const fmtflags std::ios_base::adjustfield  [static], [inherited]
```

A mask of left|right|internal. Useful for the 2-arg form of `setf`.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::internal(), std::left(), and std::right().

app

```
const openmode std::ios_base::app  [static], [inherited]
```

Seek to end before each write.
Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

ate

```
const openmode std::ios_base::ate  [static], [inherited]
```

Open and seek to end immediately after opening.
Referenced by std::basic_filebuf< _CharT, _Traits >::open().

badbit

```
const iostate std::ios_base::badbit  [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< char_type, traits_type >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::basic_ios< _CharT, _Traits >::clear(), std::basic_ostream< _CharT, _Trai std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_ostream< _CharT, _Traits >::opera std::basic_ostream< char_type, traits_type >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >: std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< char_type, traits_ty std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_istream< char_type, traits_type and std::ws().

basefield

```
const fmtflags std::ios_base::basefield  [static], [inherited]
```

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.
Referenced by std::dec(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::hex(), and std::oct().

beg

const seekdir std::ios_base::beg [static], [inherited]
Request a seek relative to the beginning of the stream.
Referenced by std::basic_filebuf< _CharT, _Traits >::seekpos().

binary

const openmode std::ios_base::binary [static], [inherited]
Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary.
Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

boolalpha

const fmtflags std::ios_base::boolalpha [static], [inherited]
Insert/extract bool in alphabetic rather than numeric format.
Referenced by std::boolalpha(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::noboolalpha().

cur

const seekdir std::ios_base::cur [static], [inherited]
Request a seek relative to the current position within the sequence.
Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

dec

const fmtflags std::ios_base::dec [static], [inherited]
Converts integer input or generates integer output in decimal base.
Referenced by std::dec().

end

const seekdir std::ios_base::end [static], [inherited]
Request a seek relative to the current end of the sequence.
Referenced by std::basic_filebuf< _CharT, _Traits >::open(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff().

eofbit

const iostate std::ios_base::eofbit [static], [inherited]
Indicates that an input operation reached the end of an input sequence.
Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >:
std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(),
std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter
std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::ge
std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< char_type, traits_type >::g
std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::putback(),
std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< char_type, traits_type >::read(), std::basic_istream< _CharT, _Traits
std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< char_type, traits_ty
std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

failbit

```
const iostate std::ios_base::failbit  [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::ge std::basic_istream< char_type, traits_type >::get(), std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::oper std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::s std::basic_ostream< _CharT, _Traits >::seekp(), and std::basic_ostream< _CharT, _Traits >::seekp().

fixed

```
const fmtflags std::ios_base::fixed  [static], [inherited]
```

Generate floating-point output in fixed-point notation.

Referenced by std::fixed(), and std::hexfloat().

floatfield

```
const fmtflags std::ios_base::floatfield  [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Referenced by std::defaultfloat(), std::fixed(), std::hexfloat(), and std::scientific().

goodbit

```
const iostate std::ios_base::goodbit  [static], [inherited]
```

Indicates all is well.

Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::d std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >: std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore() std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ostream< _CharT, _Traits > std::basic_ostream< char_type, traits_type >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >:: std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits std::basic_istream< char_type, traits_type >::seekg(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Trait and std::ws().

hex

```
const fmtflags std::ios_base::hex  [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::hex().

in

```
const openmode std::ios_base::in  [static], [inherited]
```

Open for input. Default for `ifstream` and fstream.

Referenced by std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_stringbuf< _CharT, _Traits, _

std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_filebuf< _CharT, _Traits >::underflow(), std::basic_stringbuf< _CharT, _Traits, _ and std::basic_filebuf< _CharT, _Traits >::xsgetn().

internal

```
const fmtflags std::ios_base::internal  [static], [inherited]
```

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.
Referenced by std::internal().

left

```
const fmtflags std::ios_base::left  [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)
Referenced by std::num_put< _CharT, _OutIter >::do_put(), and std::left().

oct

```
const fmtflags std::ios_base::oct  [static], [inherited]
```

Converts integer input or generates integer output in octal base.
Referenced by std::oct().

out

```
const openmode std::ios_base::out  [static], [inherited]
```

Open for output. Default for `ofstream` and fstream.
Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_stringbuf< _CharT, _Traits, std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

right

```
const fmtflags std::ios_base::right  [static], [inherited]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)
Referenced by std::right().

scientific

```
const fmtflags std::ios_base::scientific  [static], [inherited]
```

Generates floating-point output in scientific notation.
Referenced by std::hexfloat(), and std::scientific().

showbase

```
const fmtflags std::ios_base::showbase  [static], [inherited]
```

Generates a prefix indicating the numeric base of generated integer output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::noshowbase(), and std::showbase().

showpoint

```
const fmtflags std::ios_base::showpoint  [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.
Referenced by std::noshowpoint(), and std::showpoint().

showpos

```
const fmtflags std::ios_base::showpos  [static], [inherited]
```
Generates a + sign in non-negative generated numeric output.
Referenced by std::noshowpos(), and std::showpos().

skipws

```
const fmtflags std::ios_base::skipws  [static], [inherited]
```
Skips leading white space before certain input operations.
Referenced by std::noskipws(), std::__detail::operator>>(), and std::skipws().

trunc

```
const openmode std::ios_base::trunc  [static], [inherited]
```
Truncate an existing stream when opening. Default for `ofstream`.

unitbuf

```
const fmtflags std::ios_base::unitbuf  [static], [inherited]
```
Flushes output after each output operation.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::nounitbuf(), and std::unitbuf().

uppercase

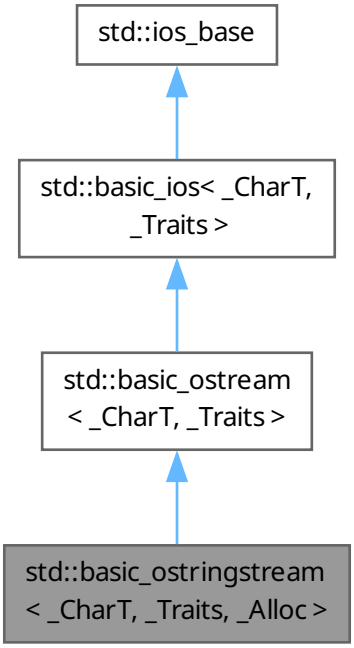
```
const fmtflags std::ios_base::uppercase  [static], [inherited]
```
Replaces certain lowercase letters with their uppercase equivalents in generated output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::nouppercase(), and std::uppercase().
The documentation for this class was generated from the following files:

- iosfwd
- ostream.h
- ostream.tcc

5.229 std::basic_ostringstream< _CharT, _Traits, _Alloc > Class Template Reference

```
#include <sstream>
```

Inheritance diagram for std::basic_ostringstream< _CharT, _Traits, _Alloc >:



Public Types

- typedef ctype< _CharT > **__ctype_type**
- typedef basic_ios< _CharT, _Traits > **__ios_type**
- typedef num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > **__num_put_type**
- typedef basic_ostream< char_type, traits_type > **__ostream_type**
- typedef basic_streambuf< _CharT, _Traits > **__streambuf_type**
- typedef basic_string< _CharT, _Traits, _Alloc > **__string_type**
- typedef basic_stringbuf< _CharT, _Traits, _Alloc > **__stringbuf_type**
- typedef _Alloc **allocator_type**
- typedef _CharT **char_type**
- enum event { **erase_event** , **imbue_event** , **copyfmt_event** }
- typedef void(* event_callback) (event __e, ios_base &__b, int __i)
- typedef traits_type::int_type **int_type**
- typedef _Ios_Iostate iostate
- typedef traits_type::off_type **off_type**
- typedef _Ios_Openmode openmode
- typedef traits_type::pos_type **pos_type**
- typedef _Ios_Seekdir seekdir
- typedef _Traits **traits_type**

- typedef num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > __num_get_type

Public Member Functions

- basic_ostringstream ()
- **basic_ostringstream** (__string_type &&__str, ios_base::openmode __mode=ios_base::out)
- **basic_ostringstream** (basic_ostringstream &&__rhs)
- basic_ostringstream (const __string_type &__str, ios_base::openmode __mode=ios_base::out)
- **basic_ostringstream** (const basic_ostringstream &)=delete
- template<typename _SAlloc>
  basic_ostringstream (const basic_string< _CharT, _Traits, _SAlloc > &__str, const allocator_type &__a)
- template<typename _SAlloc>
  basic_ostringstream (const basic_string< _CharT, _Traits, _SAlloc > &__str, ios_base::openmode __mode, const allocator_type &__a)
- template<typename _SAlloc>
  basic_ostringstream (const basic_string< _CharT, _Traits, _SAlloc > &__str, ios_base::openmode __↩ mode=ios_base::out)
- basic_ostringstream (ios_base::openmode __mode)
- **basic_ostringstream** (ios_base::openmode __mode, const allocator_type &__a)
- ~basic_ostringstream ()
- const locale & _M_getloc () const
- template<typename _ValueT>
  basic_ostream< _CharT, _Traits > & **_M_insert** (_ValueT __v)
- void **_M_setstate** (iostate __state)
- bool bad () const
- void clear (iostate __state=goodbit)
- basic_ios & copyfmt (const basic_ios &__rhs)
- bool eof () const
- iostate exceptions () const
- void exceptions (iostate __except)
- bool fail () const
- char_type fill () const
- char_type fill (char_type __ch)
- fmtflags flags () const
- fmtflags flags (fmtflags __fmtfl)
- __ostream_type & flush ()
- locale getloc () const
- bool good () const
- locale imbue (const locale &__loc)
- long & iword (int __ix)
- char narrow (char_type __c, char __dfault) const
- __ostream_type & operator<< (__streambuf_type *__sb)
- __ostream_type & operator<< (const void *__p)
- __ostream_type & **operator**<< (nullptr_t)
- basic_ostringstream & **operator=** (basic_ostringstream &&__rhs)
- basic_ostringstream & **operator=** (const basic_ostringstream &)=delete
- streamsize precision () const
- streamsize precision (streamsize __prec)
- void *& pword (int __ix)
- basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > *__sb)
- __stringbuf_type * rdbuf () const
- iostate rdstate () const
- void register_callback (event_callback __fn, int __index)
- __ostream_type & seekp (off_type, ios_base::seekdir)

- __ostream_type & seekp (pos_type)
- fmtflags setf (fmtflags __fmtfl)
- fmtflags setf (fmtflags __fmtfl, fmtflags __mask)
- void setstate (iostate __state)
- __string_type **str** () &&
- __string_type str () const &
- void **str** (__string_type &&__s)
- void str (const __string_type &__s)
- template<__allocator_like _SAlloc>
  basic_string< _CharT, _Traits, _SAlloc > **str** (const _SAlloc &__sa) const
- template<__allocator_like _SAlloc>
  requires (!is_same_v<_SAlloc, _Alloc>)
  void **str** (const basic_string< _CharT, _Traits, _SAlloc > &__s)
- void **swap** (basic_ostringstream &__rhs)
- pos_type tellp ()
- basic_ostream< _CharT, _Traits > * tie () const
- basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > *__tiestr)
- void unsetf (fmtflags __mask)
- basic_string_view< char_type, traits_type > **view** () const noexcept
- char_type widen (char __c) const
- streamsize width () const
- streamsize width (streamsize __wide)

- __ostream_type & operator<< (__ostream_type &(*__pf)(__ostream_type &))
- __ostream_type & operator<< (__ios_type &(*__pf)(__ios_type &))
- __ostream_type & operator<< (ios_base &(*__pf)(ios_base &))

Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type std::basic_ostream::sentry. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, ios_base::badbit will be turned on in the stream's error state without causing an ios_base::failure to be thrown. The original exception will then be rethrown.

- __ostream_type & operator<< (long __n)
- __ostream_type & operator<< (unsigned long __n)
- __ostream_type & operator<< (bool __n)
- __ostream_type & operator<< (short __n)
- __ostream_type & operator<< (unsigned short __n)
- __ostream_type & operator<< (int __n)
- __ostream_type & operator<< (unsigned int __n)
- __ostream_type & operator<< (long long __n)
- __ostream_type & operator<< (unsigned long long __n)

- __ostream_type & operator<< (double __f)
- __ostream_type & operator<< (float __f)
- __ostream_type & operator<< (long double __f)

Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type std::basic_ostream::sentry. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, ios_base::badbit will be turned on in the stream's error state. If badbit is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- __ostream_type & put (char_type __c)
- __ostream_type & write (const char_type *__s, streamsize __n)

- operator bool () const
- bool operator! () const

Static Public Member Functions

- static bool sync_with_stdio (bool __sync=true)
- static int xalloc () throw ()

Public Attributes

- class __attribute((__abi_tag__("cxx11"))) failure typedef _Ios_Fmtflags fmtflags

Static Public Attributes

- static const openmode **__noreplace**
- static const fmtflags adjustfield
- static const openmode app
- static const openmode ate
- static const iostate badbit
- static const fmtflags basefield
- static const seekdir beg
- static const openmode binary
- static const fmtflags boolalpha
- static const seekdir cur
- static const fmtflags dec
- static const seekdir end
- static const iostate eofbit
- static const iostate failbit
- static const fmtflags fixed
- static const fmtflags floatfield
- static const iostate goodbit
- static const fmtflags hex
- static const openmode in
- static const fmtflags internal
- static const fmtflags left
- static const fmtflags oct
- static const openmode out
- static const fmtflags right
- static const fmtflags scientific
- static const fmtflags showbase
- static const fmtflags showpoint

- static const fmtflags showpos
- static const fmtflags skipws
- static const openmode trunc
- static const fmtflags unitbuf
- static const fmtflags uppercase

Protected Types

- enum { **_S_local_word_size** }

Protected Member Functions

- void **_M_cache_locale** (const locale &__loc)
- void **_M_call_callbacks** (event __ev) throw ()
- void **_M_dispose_callbacks** (void) throw ()
- _Words & **_M_grow_words** (int __index, bool __iword)
- void **_M_init** () throw ()
- template<typename _ValueT>
  __ostream_type & **_M_insert** (_ValueT __v)
- void **_M_move** (ios_base &) noexcept
- void **_M_swap** (ios_base &__rhs) noexcept
- void init (basic_streambuf< _CharT, _Traits > *__sb)
- void **move** (basic_ios &&__rhs)
- void **move** (basic_ios &__rhs)
- void **set_rdbuf** (basic_streambuf< _CharT, _Traits > *__sb)
- void **swap** (basic_ios &__rhs) noexcept
- void **swap** (basic_ostream &__rhs)

Protected Attributes

- _Callback_list * **_M_callbacks**
- const __ctype_type * **_M_ctype**
- iostate **_M_exception**
- char_type **_M_fill**
- bool **_M_fill_init**
- fmtflags **_M_flags**
- locale **_M_ios_locale**
- _Words **_M_local_word** [_S_local_word_size]
- const __num_get_type * **_M_num_get**
- const __num_put_type * **_M_num_put**
- streamsize **_M_precision**
- basic_streambuf< _CharT, _Traits > * **_M_streambuf**
- iostate **_M_streambuf_state**
- basic_ostream< _CharT, _Traits > * **_M_tie**
- streamsize **_M_width**
- _Words * **_M_word**
- int **_M_word_size**
- _Words **_M_word_zero**

5.229.1 Detailed Description

template<typename _CharT, typename _Traits, typename _Alloc>
class std::basic_ostringstream< _CharT, _Traits, _Alloc >

Controlling output for std::string.

Template Parameters

_CharT	Type of character stream.
_Traits	Traits for character type, defaults to char_traits<_CharT>.
_Alloc	Allocator type, defaults to allocator<_CharT>.

This class supports writing to objects of type std::basic_string, using the inherited functions from std::basic_ostream. To control the associated sequence, an instance of std::basic_stringbuf is used, which this page refers to as `sb`.

5.229.2 Member Typedef Documentation

__num_get_type

```
template<typename _CharT, typename _Traits>
typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits> > std::basic_ios< _CharT, _Traits
>::__num_get_type  [inherited]
```

These are non-standard types.

event_callback

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i)  [inherited]
```

The type of an event callback function.

Parameters

_e	One of the members of the event enum.
_b	Reference to the ios_base object.
_i	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several ios_base and basic_ios functions, specifically imbue(), copyfmt(), and ~ios().

iostate

```
typedef _Ios_Iostate std::ios_base::iostate  [inherited]
```

This is a bitmask type.
_Ios_Iostate is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit
- eofbit
- failbit
- goodbit

openmode

```
typedef _Ios_Openmode std::ios_base::openmode  [inherited]
```

This is a bitmask type.
_Ios_Openmode is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type openmode are:

- app
- ate
- binary
- in
- out
- trunc

seekdir

`typedef _Ios_Seekdir std::ios_base::seekdir [inherited]`
This is an enumerated type.
`_Ios_Seekdir` `is` implementation-defined. Defined values of type seekdir are:

- beg
- cur, equivalent to `SEEK_CUR` in the C standard library.
- end, equivalent to `SEEK_END` in the C standard library.

5.229.3 Member Enumeration Documentation

event

`enum std::ios_base::event [inherited]`
The set of events that may be passed to an event callback.
erase_event is used during ~ios() and copyfmt(). imbue_event is used during imbue(). copyfmt_event is used during copyfmt().

5.229.4 Constructor & Destructor Documentation

basic_ostringstream() [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_ostringstream< _CharT, _Traits, _Alloc >::basic_ostringstream () [inline]
```
Default constructor starts with an empty string buffer.
Initializes `sb` using `mode|out`, and passes `&sb` to the base class initializer. Does not allocate any buffer.
That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

basic_ostringstream() [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_ostringstream< _CharT, _Traits, _Alloc >::basic_ostringstream (
            ios_base::openmode __mode) [inline], [explicit]
```
Starts with an empty string buffer.

Parameters

__mode	Whether the buffer can read, or write, or both.

`ios_base::out` is automatically included in *mode*.
Initializes `sb` using `mode|out`, and passes `&sb` to the base class initializer. Does not allocate any buffer.
That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

basic_ostringstream() [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_ostringstream< _CharT, _Traits, _Alloc >::basic_ostringstream (
            const __string_type & __str,
            ios_base::openmode __mode = ios_base::out)  [inline], [explicit]
```

Starts with an existing string buffer.

Parameters

__str	A string to copy as a starting buffer.
__mode	Whether the buffer can read, or write, or both.

`ios_base::out` is automatically included in *mode*.
Initializes `sb` using *str* and `mode|out`, and passes `&sb` to the base class initializer.
That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

~basic_ostringstream()

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_ostringstream< _CharT, _Traits, _Alloc >::~basic_ostringstream ()  [inline]
```

The destructor does nothing.
The buffer is deallocated by the stringbuf object, not the formatting stream.

5.229.5 Member Function Documentation

_M_getloc()

```
const locale & std::ios_base::_M_getloc () const  [inline], [inherited]
```

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.
Referenced by std::money_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter
std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIte
std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::num_put< _CharT, _OutIter
std::time_put< _CharT, _OutIter >::do_put(), std::time_get< _CharT, _InIter >::get(), and std::time_put< _CharT, _OutIter >::put().

bad()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry().

clear()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::clear (
             iostate __state = goodbit)  [inherited]
```

[Re]sets the error state.

Parameters

__state	The new state flag(s) to set.

See std::ios_base::iostate for the possible bit values. Most users will not need to pass an argument.
References std::ios_base::badbit, exceptions(), rdbuf(), and rdstate().
Referenced by std::basic_ios< char_type, traits_type >::exceptions(), std::__detail::operator>>(), std::basic_istream< _CharT, _Traits >
rdbuf(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char_type, traits
and std::basic_istream< _CharT, _Traits >::unget().

copyfmt()

```
template<typename _CharT, typename _Traits>
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
            const basic_ios< _CharT, _Traits > & __rhs)  [inherited]
```

Copies fields of __rhs into this.

Parameters

__rhs	The source values for the copies.

Returns

Reference to this object.

All fields of __rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase_event. After copying, each (new) callback is invoked with copyfmt_event. The final step is to copy exceptions().
References basic_ios(), std::__addressof(), exceptions(), fill(), std::ios_base::flags(), std::ios_base::getloc(), std::ios_base::precision(), tie(), std::tie(), and std::ios_base::width().

eof()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

exceptions() [1/2]

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions () const  [inline], [nodiscard], [inherited]
```

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.
Referenced by clear(), and copyfmt().

exceptions() [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
            iostate __except)  [inline], [inherited]
```

Throwing exceptions on errors.

Parameters

__except	The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type std::ios_base::failure is thrown.
If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr « "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr « "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.
Referenced by std::basic_ios< char_type, traits_type >::operator bool(), std::basic_ios< char_type, traits_type >::operator!(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >:: std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >:: and std::regex_traits< _Ch_type >::value().

fill() [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const  [inline], [nodiscard], [inherited]
```

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.
Referenced by copyfmt().

fill() [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
            char_type __ch)  [inline], [inherited]
```

Sets a new *empty* character.

Parameters

__ch	The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

flags() [1/2]

```
fmtflags std::ios_base::flags () const  [inline], [nodiscard], [inherited]
```

Access to format flags.

Returns

The format control flags for both input and output.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >
std::num_put< _CharT, _OutIter >::do_put(), std::num_put< _CharT, _OutIter >::do_put(), std::chrono::operator<<(),
std::operator<<(), std::__detail::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(),
std::operator>>(), std::operator>>(), and std::operator>>().

flags() [2/2]

```
fmtflags std::ios_base::flags (
            fmtflags __fmtfl)  [inline], [inherited]
```

Setting new format flags all at once.

Parameters

__fmtfl	The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *__fmtfl*.
References fmtflags.

flush()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::flush ()  [inherited]
```

Synchronizing the stream buffer.

Returns

*this

If `rdbuf()` is a null pointer, changes nothing.
Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit.
References std::ios_base::badbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits

getloc()

```
locale std::ios_base::getloc () const  [inline], [nodiscard], [inherited]
```

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::money_put< _CharT, _OutIter >::do_put(), std::basic_ios< _CharT, _Tr
std::chrono::operator<<(), std::operator>>(), std::operator>>(), and std::ws().

good()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if no error flags are set.

A wrapper around rdstate.
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::__detail::operator>>().

imbue()

```
template<typename _CharT, typename _Traits>
locale std::basic_ios< _CharT, _Traits >::imbue (
            const locale & __loc)  [inherited]
```

Moves to a new locale.

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.
Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html
References std::ios_base::getloc(), std::ios_base::imbue(), and rdbuf().
Referenced by std::chrono::operator<<().

init()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::init (
            basic_streambuf< _CharT, _Traits > * __sb)  [protected], [inherited]
```

All setup is performed here.
This is called from the public constructor. It is not virtual and cannot be redefined.
Referenced by std::basic_ios< char_type, traits_type >::basic_ios(), and std::basic_ostream< char_type, traits_type >::basic_ostream().

iword()

```
long & std::ios_base::iword (
            int __ix)  [inline], [inherited]
```

Access to integer array.

Parameters

_↩ _ix	Index into the array.

Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.
References iword().
Referenced by iword().

narrow()

```
template<typename _CharT, typename _Traits>
char std::basic_ios< _CharT, _Traits >::narrow (
            char_type __c,
            char __dfault) const  [inline], [inherited]
```

Squeezes characters.

Parameters

__c	The character to narrow.
__dfault	The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.
Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c,dfault)
```

Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.↩ html

operator bool()

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool () const  [inline], [explicit], [nodiscard],
[inherited]
```

The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

operator"!()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator!  () const  [inline], [nodiscard], [inherited]
```

The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

operator<<() [1/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            __ios_type &(* __pf )(__ios_type &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

operator<<() [2/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            __ostream_type &(* __pf )(__ostream_type &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

operator<<() [3/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
            __streambuf_type * __sb)  [inherited]
```

Extracting from another streambuf.

Parameters

__sb	A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.
If `__sb` is NULL, the stream will set failbit in its error state.
Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets failbit in the error state

If the function inserts no characters, failbit is set.

operator<<() [4/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            bool __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [5/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            const void * __p)  [inline], [inherited]
```

Pointer arithmetic inserters.

Parameters

_↩ _p	A variable of pointer type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [6/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            double __f)  [inline], [inherited]
```

Floating point arithmetic inserters.

Parameters

↩ _↩ ↩ _↩ f	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [7/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            float __f)  [inline], [inherited]
```

Floating point arithmetic inserters.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [8/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
            int __n)  [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _*n*	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [9/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            ios_base &(* __pf )(ios_base &))  [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

operator<<() [10/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            long __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _*n*	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [11/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            long double __f)  [inline], [inherited]
```

Floating point arithmetic inserters.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [12/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            long long __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ *_n*	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [13/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
            short __n)  [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ *_n*	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.
References basic_ostream(), std::ios_base::badbit, std::ostreambuf_iterator< _CharT, _Traits >::failed(), std::ios_base::goodbit, std::num_put< _CharT, _OutIter >::put(), std::basic_ios< _CharT, _Traits >::setstate(), and std::use_facet().

operator<<() [14/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned int __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [15/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned long __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [16/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned long long __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [17/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned short __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

precision() [1/2]

```
streamsize std::ios_base::precision () const  [inline], [nodiscard], [inherited]
```

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::chrono::operator<<().

precision() [2/2]

```
streamsize std::ios_base::precision (
            streamsize __prec)  [inline], [inherited]
```

Changing flags.

Parameters

__prec	The new precision value.

Returns

The previous value of precision().

put()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (
            char_type __c)  [inherited]
```

Simple insertion.

Parameters

_↩ _c	The character to insert.

Returns

*this

Tries to insert __c.

Note

This function is not overloaded on signed char and unsigned char.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits

pword()

```
void *& std::ios_base::pword (
            int __ix)  [inline], [inherited]
```

Access to void pointer array.

Parameters

_↩ _ix	Index into the array.

Returns

A reference to a void* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

rdbuf() [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
            basic_streambuf< _CharT, _Traits > * __sb)  [inherited]
```

Changing the underlying buffer.

Parameters

__sb	The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.
Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument rdbuf(), which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream     foo;         // or some other derived type
std::streambuf*  p = .....;

foo.ios::rdbuf(p);            // ios == basic_ios<char>
```

References clear().

rdbuf() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
__stringbuf_type * std::basic_ostringstream< _CharT, _Traits, _Alloc >::rdbuf () const  [inline],
[nodiscard]
```

Accessing the underlying buffer.

Returns

The current basic_stringbuf buffer.

This hides both signatures of std::basic_ios::rdbuf().

rdstate()

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::rdstate () const  [inline], [nodiscard], [inherited]
```

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See std::ios_base::iostate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().
Referenced by std::basic_ios< char_type, traits_type >::bad(), clear(), std::basic_ios< char_type, traits_type >::eof(), std::basic_ios< char_type, traits_type >::fail(), std::basic_ios< char_type, traits_type >::good(), std::basic_istream< _CharT, _Traits >::
std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char_type, traits_type >
and std::basic_istream< _CharT, _Traits >::unget().

register_callback()

```
void std::ios_base::register_callback (
            event_callback __fn,
            int __index)  [inherited]
```

Add the callback __fn with parameter __index.

Parameters

__fn	The function to add.
__index	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.
References fmtflags.

seekp() [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
            off_type __off,
            ios_base::seekdir __dir)  [inherited]
```

Changing the current write position.

Parameters

__off	A file offset object.
__dir	The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off,dir)`. If that function fails, sets failbit.
References std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::out, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

seekp() [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
            pos_type __pos)  [inherited]
```

Changing the current write position.

Parameters

__pos	A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.
References std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::out, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

setf() [1/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl)  [inline], [inherited]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.
References fmtflags.
Referenced by std::__detail::operator>>().

setf() [2/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl,
            fmtflags __mask)  [inline], [inherited]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.
__mask	The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.
References fmtflags.

setstate()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
            iostate __state)  [inline], [inherited]
```

Sets additional flags in the error state.

Parameters

__state	The additional state flag(s) to set.

See std::ios_base::iostate for the possible bit values.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::getline(), std::getline(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _Cha std::operator>>(), std::operator>>(), std::operator>>(), std::tr2::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >: std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits > std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), and std::ws().

str() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
__string_type std::basic_ostringstream< _CharT, _Traits, _Alloc >::str () const &  [inline],
[nodiscard]
```

Copying out the string buffer.

Returns

```
rdbuf()->str()
```

str() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_ostringstream< _CharT, _Traits, _Alloc >::str (
            const __string_type & __s)  [inline]
```

Setting a new buffer.

Parameters

__s	The string to use as a new sequence.

Calls `rdbuf()->str(s)`.

sync_with_stdio()

```
static bool std::ios_base::sync_with_stdio (
            bool __sync = true)  [static], [inherited]
```

Interaction with the standard C I/O objects.

Parameters

__sync	Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary

tellp()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits >::pos_type std::basic_ostream< _CharT, _Traits >::tellp ()  [inherited]
```

Getting the current write position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type`(-1) to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0,cur,out)`.
References std::ios_base::cur, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::out, and std::basic_ios< _CharT, _Traits >::rdbuf()

tie() [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie () const  [inline],
[nodiscard], [inherited]
```

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and copyfmt().

tie() [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie (
            basic_ostream< _CharT, _Traits > * __tiestr)  [inline], [inherited]
```

Ties this stream to an output stream.

Parameters

__tiestr	The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

unsetf()

```
void std::ios_base::unsetf (
            fmtflags __mask)  [inline], [inherited]
```

Clearing format flags.

Parameters

__mask	The flags to unset.

This function clears *__mask* in the format flags.
References fmtflags.

widen()

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
            char __c) const  [inline], [inherited]
```

Widens characters.

Parameters

_c	The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.
Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html
Referenced by std::basic_ios< char_type, traits_type >::fill(), std::getline(), std::getline(), and std::tr2::operator>>().

width() [1/2]

```
streamsize std::ios_base::width () const  [inline], [nodiscard], [inherited]
```

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_put< _CharT, _OutIter >::do_put(), std::operator>>(), and std::operator>>().

width() [2/2]

```
streamsize std::ios_base::width (
            streamsize __wide)  [inline], [inherited]
```

Changing flags.

Parameters

__wide	The new width value.

Returns

The previous value of width().

write()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::write (
            const char_type * __s,
            streamsize __n)  [inherited]
```

Character string insertion.

Parameters

_↩ _s	The array to insert.
_↩ _n	Maximum number of characters to insert.

Returns

*this

Characters are copied from __s and inserted into the stream until one of the following happens:

- __n characters are inserted
- inserting into the output sequence fails (in this case, badbit will be set in the stream's error state)

Note

This function is not overloaded on signed char and unsigned char.

xalloc()

```
static int std::ios_base::xalloc () throw ()   [static], [inherited]
```

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the iword and pword functions. The expectation is that an application calls xalloc in order to obtain an index in the iword and pword arrays that can be used without fear of conflict.
The implementation maintains a static variable that is incremented and returned on each invocation. xalloc is guaranteed to return an index that is safe to use in the iword and pword arrays.

5.229.6 Member Data Documentation

adjustfield

```
const fmtflags std::ios_base::adjustfield  [static], [inherited]
```

A mask of left|right|internal. Useful for the 2-arg form of `setf`.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::internal(), std::left(), and std::right().

app

```
const openmode std::ios_base::app  [static], [inherited]
```

Seek to end before each write.
Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

ate

```
const openmode std::ios_base::ate  [static], [inherited]
```

Open and seek to end immediately after opening.
Referenced by std::basic_filebuf< _CharT, _Traits >::open().

badbit

```
const iostate std::ios_base::badbit  [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< char_type, traits_type >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::basic_ios< _CharT, _Traits >::clear(), std::basic_ostream< _CharT, _Trai std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_ostream< _CharT, _Traits >::opera std::basic_ostream< char_type, traits_type >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >: std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< char_type, traits_ty std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_istream< char_type, traits_type and std::ws().

basefield

```
const fmtflags std::ios_base::basefield  [static], [inherited]
```

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.
Referenced by std::dec(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::hex(), and std::oct().

beg

```
const seekdir std::ios_base::beg  [static], [inherited]
```

Request a seek relative to the beginning of the stream.
Referenced by std::basic_filebuf< _CharT, _Traits >::seekpos().

binary

```
const openmode std::ios_base::binary  [static], [inherited]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.↩filestreams.binary.
Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

boolalpha

```
const fmtflags std::ios_base::boolalpha [static], [inherited]
```

Insert/extract `bool` in alphabetic rather than numeric format.
Referenced by std::boolalpha(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::noboolalpha().

cur

```
const seekdir std::ios_base::cur [static], [inherited]
```

Request a seek relative to the current position within the sequence.
Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

dec

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.
Referenced by std::dec().

end

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.
Referenced by std::basic_filebuf< _CharT, _Traits >::open(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff().

eofbit

```
const iostate std::ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.
Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >:: std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::ge std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< char_type, traits_type >::g std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< char_type, traits_type >::read(), std::basic_istream< _CharT, _Traits std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< char_type, traits_ty std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

failbit

```
const iostate std::ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::ge std::basic_istream< char_type, traits_type >::get(), std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::oper std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::s std::basic_ostream< _CharT, _Traits >::seekp(), and std::basic_ostream< _CharT, _Traits >::seekp().

fixed

```
const fmtflags std::ios_base::fixed  [static], [inherited]
```
Generate floating-point output in fixed-point notation.
Referenced by std::fixed(), and std::hexfloat().

floatfield

```
const fmtflags std::ios_base::floatfield  [static], [inherited]
```
A mask of scientific|fixed. Useful for the 2-arg form of `setf`.
Referenced by std::defaultfloat(), std::fixed(), std::hexfloat(), and std::scientific().

goodbit

```
const iostate std::ios_base::goodbit  [static], [inherited]
```
Indicates all is well.
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::d std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >: std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore() std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ostream< _CharT, _Traits > std::basic_ostream< char_type, traits_type >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >:: std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits std::basic_istream< char_type, traits_type >::seekg(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Trait and std::ws().

hex

```
const fmtflags std::ios_base::hex  [static], [inherited]
```
Converts integer input or generates integer output in hexadecimal base.
Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::hex().

in

```
const openmode std::ios_base::in  [static], [inherited]
```
Open for input. Default for `ifstream` and fstream.
Referenced by std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_stringbuf< _CharT, _Traits, _ std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_filebuf< _CharT, _Traits >::underflow(), std::basic_stringbuf< _CharT, _Traits, _ and std::basic_filebuf< _CharT, _Traits >::xsgetn().

internal

```
const fmtflags std::ios_base::internal  [static], [inherited]
```
Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.
Referenced by std::internal().

left

```
const fmtflags std::ios_base::left  [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)
Referenced by std::num_put< _CharT, _OutIter >::do_put(), and std::left().

oct

```
const fmtflags std::ios_base::oct  [static], [inherited]
```
Converts integer input or generates integer output in octal base.
Referenced by std::oct().

out

```
const openmode std::ios_base::out  [static], [inherited]
```
Open for output. Default for `ofstream` and fstream.
Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_stringbuf< _CharT, _Traits, std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

right

```
const fmtflags std::ios_base::right  [static], [inherited]
```
Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)
Referenced by std::right().

scientific

```
const fmtflags std::ios_base::scientific  [static], [inherited]
```
Generates floating-point output in scientific notation.
Referenced by std::hexfloat(), and std::scientific().

showbase

```
const fmtflags std::ios_base::showbase  [static], [inherited]
```
Generates a prefix indicating the numeric base of generated integer output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::noshowbase(), and std::showbase().

showpoint

```
const fmtflags std::ios_base::showpoint  [static], [inherited]
```
Generates a decimal-point character unconditionally in generated floating-point output.
Referenced by std::noshowpoint(), and std::showpoint().

showpos

```
const fmtflags std::ios_base::showpos  [static], [inherited]
```
Generates a + sign in non-negative generated numeric output.
Referenced by std::noshowpos(), and std::showpos().

skipws

```
const fmtflags std::ios_base::skipws  [static], [inherited]
```
Skips leading white space before certain input operations.
Referenced by std::noskipws(), std::__detail::operator>>(), and std::skipws().

trunc

```
const openmode std::ios_base::trunc  [static], [inherited]
```
Truncate an existing stream when opening. Default for `ofstream`.

unitbuf

```
const fmtflags std::ios_base::unitbuf  [static], [inherited]
```
Flushes output after each output operation.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::nounitbuf(), and std::unitbuf().

uppercase

```
const fmtflags std::ios_base::uppercase  [static], [inherited]
```
Replaces certain lowercase letters with their uppercase equivalents in generated output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::nouppercase(), and std::uppercase().
The documentation for this class was generated from the following files:

- iosfwd
- sstream

5.230 std::basic_regex< _Ch_type, _Rx_traits > Class Template Reference

```
#include <regex>
```

Public Types

- typedef regex_constants::syntax_option_type **flag_type**
- typedef traits_type::locale_type **locale_type**
- typedef traits_type::string_type **string_type**
- typedef _Rx_traits **traits_type**
- typedef _Ch_type **value_type**

Public Member Functions

- basic_regex () noexcept
- template<typename _FwdIter>
  basic_regex (_FwdIter __first, _FwdIter __last, flag_type __f=ECMAScript)
- basic_regex (basic_regex &&__rhs) noexcept=default
- basic_regex (const _Ch_type *__p, flag_type __f=ECMAScript)
- basic_regex (const _Ch_type *__p, std::size_t __len, flag_type __f=ECMAScript)
- basic_regex (const basic_regex &__rhs)=default
- template<typename _Ch_traits, typename _Ch_alloc>
  basic_regex (const std::basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, flag_type __f=ECMAScript)
- basic_regex (initializer_list< _Ch_type > __l, flag_type __f=ECMAScript)
- ~basic_regex ()
- template<typename _InputIterator>
  basic_regex & assign (_InputIterator __first, _InputIterator __last, flag_type __flags=ECMAScript)
- basic_regex & assign (basic_regex &&__rhs) noexcept
- basic_regex & assign (const _Ch_type *__p, flag_type __flags=ECMAScript)
- basic_regex & assign (const _Ch_type *__p, size_t __len, flag_type __flags=ECMAScript)
- basic_regex & assign (const basic_regex &__rhs) noexcept
- template<typename _Ch_traits, typename _Alloc>
  basic_regex & assign (const basic_string< _Ch_type, _Ch_traits, _Alloc > &__s, flag_type __flags=ECMAScript)

- basic_regex & assign (initializer_list< _Ch_type > __l, flag_type __flags=ECMAScript)
- flag_type flags () const noexcept
- locale_type getloc () const noexcept
- locale_type imbue (locale_type __loc)
- unsigned int mark_count () const noexcept
- basic_regex & operator= (basic_regex &&)=default
- basic_regex & operator= (const _Ch_type *__p)
- basic_regex & operator= (const basic_regex &)=default
- template<typename _Ch_traits, typename _Alloc>
  basic_regex & operator= (const basic_string< _Ch_type, _Ch_traits, _Alloc > &__s)
- basic_regex & operator= (initializer_list< _Ch_type > __l)
- void swap (basic_regex &__rhs) noexcept

Static Public Attributes

Constants

std [28.8.1](1)

- static constexpr flag_type **icase**
- static constexpr flag_type **nosubs**
- static constexpr flag_type **optimize**
- static constexpr flag_type **collate**
- static constexpr flag_type **ECMAScript**
- static constexpr flag_type **basic**
- static constexpr flag_type **extended**
- static constexpr flag_type **awk**
- static constexpr flag_type **grep**
- static constexpr flag_type **egrep**
- static constexpr flag_type **multiline**

Friends

- template<typename _Bp, typename _Ap, typename _Cp, typename _Rp>
  bool **__detail::__regex_algo_impl** (_Bp, _Bp, match_results< _Bp, _Ap > &, const basic_regex< _Cp, _Rp > &, regex_constants::match_flag_type, __detail::_RegexExecutorPolicy, bool)
- template<typename, typename, typename, bool>
  class **__detail::_Executor**

Related Symbols

(Note that these are not member symbols.)

- template<typename _Ch_type, typename _Rx_traits>
  void swap (basic_regex< _Ch_type, _Rx_traits > &__lhs, basic_regex< _Ch_type, _Rx_traits > &__rhs) noexcept

5.230.1 Detailed Description

template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
class std::basic_regex< _Ch_type, _Rx_traits >

A regular expression.
Specializations of this class template represent regular expressions constructed from sequences of character type `_Ch_type`. Use the `std::regex` typedef for `std::basic_regex<char>`.

A character sequence passed to the constructor will be parsed according to the chosen grammar, and used to create a state machine representing the regular expression. The regex object can then be passed to algorithms such as `std::regex_match` to match sequences of characters.
The `syntax_option_type` flag passed to the constructor selects from one of the supported regular expression grammars. The default is `ECMAScript` and the others are `basic`, `extended`, `awk`, `grep`, and `egrep`, which are variations on POSIX regular expressions.

Since

C++11

5.230.2 Constructor & Destructor Documentation

basic_regex() [1/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex ()  [inline], [noexcept]
```

Constructs a basic regular expression that does not match any character sequence.

basic_regex() [2/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (
            const _Ch_type * __p,
            flag_type __f = ECMAScript)  [inline], [explicit]
```

Constructs a basic regular expression from the sequence [__p, __p + char_traits<_Ch_type>::length(__p)) interpreted according to the flags in __f.

Parameters

__p	A pointer to the start of a C-style null-terminated string containing a regular expression.
__f	Flags indicating the syntax rules and options.

Exceptions

regex_error	if __p is not a valid regular expression.

basic_regex() [3/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (
            const _Ch_type * __p,
            std::size_t __len,
            flag_type __f = ECMAScript)  [inline]
```

Constructs a basic regular expression from the sequence [p, p + len) interpreted according to the flags in `f`.

Parameters

__p	A pointer to the start of a string containing a regular expression.
__len	The length of the string containing the regular expression.
__f	Flags indicating the syntax rules and options.

Exceptions

regex_error	if __p is not a valid regular expression.

basic_regex() [4/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (
            const basic_regex< _Ch_type, _Rx_traits > & __rhs)  [default]
```

Copy-constructs a basic regular expression.

Parameters

__rhs	A regex object.

basic_regex() [5/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (
            basic_regex< _Ch_type, _Rx_traits > && __rhs)  [default], [noexcept]
```

Move-constructs a basic regular expression.

Parameters

__rhs	A regex object.

basic_regex() [6/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
template<typename _Ch_traits, typename _Ch_alloc>
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (
            const std::basic_string< _Ch_type, _Ch_traits, _Ch_alloc > & __s,
            flag_type __f = ECMAScript)  [inline], [explicit]
```

Constructs a basic regular expression from the string s interpreted according to the flags in f.

Parameters

_↩ _s	A string containing a regular expression.
_↩ _f	Flags indicating the syntax rules and options.

Exceptions

regex_error	if __s is not a valid regular expression.

basic_regex() [7/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
template<typename _FwdIter>
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (
            _FwdIter __first,
            _FwdIter __last,
            flag_type __f = ECMAScript)  [inline]
```

Constructs a basic regular expression from the range [first, last) interpreted according to the flags in `f`.

Parameters

__first	The start of a range containing a valid regular expression.
__last	The end of a range containing a valid regular expression.
__f	The format flags of the regular expression.

Exceptions

regex_error	if [__first, __last) is not a valid regular expression.

basic_regex() [8/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (
            initializer_list< _Ch_type > __l,
            flag_type __f = ECMAScript)  [inline]
```

Constructs a basic regular expression from an initializer list.

Parameters

↩ _↩ ↩ _↩ *l*	The initializer list.
↩ _↩ ↩ _↩ *f*	The format flags of the regular expression.

Exceptions

regex_error	if `__l` is not a valid regular expression.

~basic_regex()

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
std::basic_regex< _Ch_type, _Rx_traits >::~basic_regex ()  [inline]
```

Destroys a basic regular expression.

5.230.3 Member Function Documentation

assign() [1/7]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
template<typename _InputIterator>
basic_regex & std::basic_regex< _Ch_type, _Rx_traits >::assign (
            _InputIterator __first,
            _InputIterator __last,
            flag_type __flags = ECMAScript)  [inline]
```

Assigns a new regular expression to a regex object.

Parameters

__first	The start of a range containing a valid regular expression.
__last	The end of a range containing a valid regular expression.
__flags	Syntax option flags.

Exceptions

regex_error	if p does not contain a valid regular expression pattern interpreted according to __flags. If regex_error is thrown, the object remains unchanged.

assign() [2/7]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex & std::basic_regex< _Ch_type, _Rx_traits >::assign (
            basic_regex< _Ch_type, _Rx_traits > && __rhs)  [inline], [noexcept]
```

Move-assigns one regular expression to another.

Parameters

__rhs	Another regular expression object.

assign() [3/7]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex & std::basic_regex< _Ch_type, _Rx_traits >::assign (
            const _Ch_type * __p,
            flag_type __flags = ECMAScript)  [inline]
```

Assigns a new regular expression to a regex object from a C-style null-terminated string containing a regular expression pattern.

Parameters

__p	A pointer to a C-style null-terminated string containing a regular expression pattern.
__flags	Syntax option flags.

Exceptions

regex_error	if __p does not contain a valid regular expression pattern interpreted according to __flags. If regex_error is thrown, *this remains unchanged.

assign() [4/7]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex & std::basic_regex< _Ch_type, _Rx_traits >::assign (
            const _Ch_type * __p,
            size_t __len,
            flag_type __flags = ECMAScript)  [inline]
```

Assigns a new regular expression to a regex object from a C-style string containing a regular expression pattern.

Parameters

__p	A pointer to a C-style string containing a regular expression pattern.
__len	The length of the regular expression pattern string.
__flags	Syntax option flags.

Exceptions

regex_error	if p does not contain a valid regular expression pattern interpreted according to __flags. If regex_error is thrown, *this remains unchanged.

assign() [5/7]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex & std::basic_regex< _Ch_type, _Rx_traits >::assign (
            const basic_regex< _Ch_type, _Rx_traits > & __rhs)  [inline], [noexcept]
```

Assigns one regular expression to another.

Parameters

__rhs	Another regular expression object.

Referenced by std::basic_regex< char >::basic_regex(), std::basic_regex< char >::assign(), std::basic_regex< char >::operator=(), std::basic_regex< char >::operator=(), and std::basic_regex< char >::operator=().

assign() [6/7]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
template<typename _Ch_traits, typename _Alloc>
basic_regex & std::basic_regex< _Ch_type, _Rx_traits >::assign (
            const basic_string< _Ch_type, _Ch_traits, _Alloc > & __s,
            flag_type __flags = ECMAScript)  [inline]
```

Assigns a new regular expression to a regex object from a string containing a regular expression pattern.

Parameters

__s	A string containing a regular expression pattern.
__flags	Syntax option flags.

Exceptions

regex_error	if __s does not contain a valid regular expression pattern interpreted according to __flags. If regex_error is thrown, *this remains unchanged.

assign() [7/7]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex & std::basic_regex< _Ch_type, _Rx_traits >::assign (
             initializer_list< _Ch_type > __l,
             flag_type __flags = ECMAScript)  [inline]
```

Assigns a new regular expression to a regex object.

Parameters

__l	An initializer list representing a regular expression.
__flags	Syntax option flags.

Exceptions

regex_error	if __l does not contain a valid regular expression pattern interpreted according to __flags. If regex_error is thrown, the object remains unchanged.

flags()

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
flag_type std::basic_regex< _Ch_type, _Rx_traits >::flags () const  [inline], [noexcept]
```

Gets the flags used to construct the regular expression or in the last call to assign().

getloc()

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
locale_type std::basic_regex< _Ch_type, _Rx_traits >::getloc () const  [inline], [noexcept]
```

Gets the locale currently imbued in the regular expression object.

imbue()

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
locale_type std::basic_regex< _Ch_type, _Rx_traits >::imbue (
             locale_type __loc)  [inline]
```

Imbues the regular expression object with the given locale.

Parameters

__loc	A locale.

mark_count()

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
unsigned int std::basic_regex< _Ch_type, _Rx_traits >::mark_count () const  [inline], [noexcept]
```

Gets the number of marked subexpressions within the regular expression.

operator=() [1/5]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex & std::basic_regex< _Ch_type, _Rx_traits >::operator= (
            basic_regex< _Ch_type, _Rx_traits > && )  [default]
```

Move-assigns one regular expression to another.

operator=() [2/5]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex & std::basic_regex< _Ch_type, _Rx_traits >::operator= (
            const _Ch_type * __p)  [inline]
```

Replaces a regular expression with a new one constructed from a C-style null-terminated string.

Parameters

_↩ _p	A pointer to the start of a null-terminated C-style string containing a regular expression.

operator=() [3/5]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex & std::basic_regex< _Ch_type, _Rx_traits >::operator= (
            const basic_regex< _Ch_type, _Rx_traits > & )  [default]
```

Assigns one regular expression to another.

operator=() [4/5]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
template<typename _Ch_traits, typename _Alloc>
basic_regex & std::basic_regex< _Ch_type, _Rx_traits >::operator= (
            const basic_string< _Ch_type, _Ch_traits, _Alloc > & __s)  [inline]
```

Replaces a regular expression with a new one constructed from a string.

Parameters

_↩ _s	A pointer to a string containing a regular expression.

operator=() [5/5]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex & std::basic_regex< _Ch_type, _Rx_traits >::operator= (
            initializer_list< _Ch_type > __l)  [inline]
```

Replaces a regular expression with a new one constructed from an initializer list.

Parameters

↩ _↩ ↩ _↩ *l*	The initializer list.

Exceptions

regex_error	if `__l` is not a valid regular expression.

swap()

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
void std::basic_regex< _Ch_type, _Rx_traits >::swap (
             basic_regex< _Ch_type, _Rx_traits > & __rhs)  [inline], [noexcept]
```

Swaps the contents of two regular expression objects.

Parameters

__rhs	Another regular expression object.

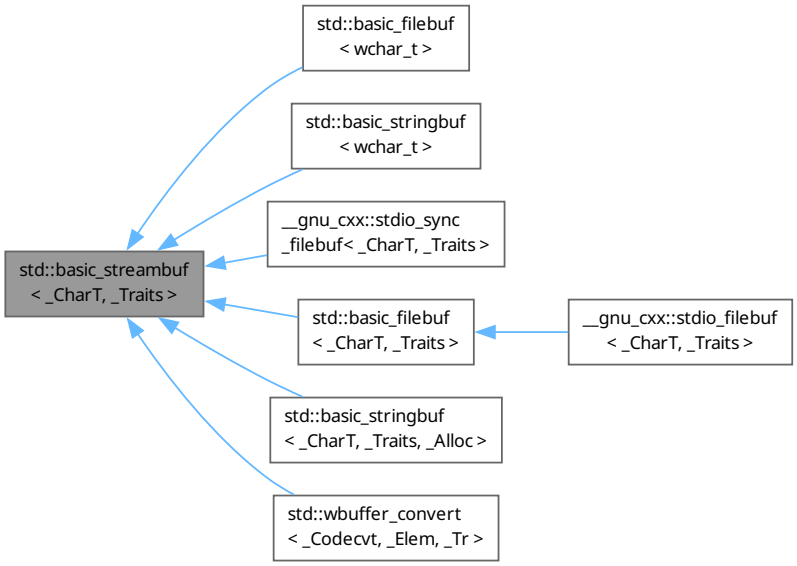
The documentation for this class was generated from the following file:

- regex.h

5.231 std::basic_streambuf< _CharT, _Traits > Class Template Reference

```
#include <streambuf>
```

Inheritance diagram for std::basic_streambuf< _CharT, _Traits >:



Public Types

- typedef _CharT char_type
- typedef _Traits traits_type
- typedef traits_type::int_type int_type
- typedef traits_type::pos_type pos_type
- typedef traits_type::off_type off_type

- typedef basic_streambuf< char_type, traits_type > __streambuf_type

Public Member Functions

- virtual ~basic_streambuf ()
- locale getloc () const
- streamsize in_avail ()
- locale pubimbue (const locale &__loc)
- int_type sbumpc ()
- int_type sgetc ()
- streamsize sgetn (char_type *__s, streamsize __n)
- int_type snextc ()
- int_type sputbackc (char_type __c)
- int_type sputc (char_type __c)
- streamsize sputn (const char_type *__s, streamsize __n)
- int_type sungetc ()

- basic_streambuf * pubsetbuf (char_type *__s, streamsize __n)
- pos_type pubseekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)
- pos_type pubseekpos (pos_type __sp, ios_base::openmode __mode=ios_base::in|ios_base::out)
- int pubsync ()

Protected Member Functions

- basic_streambuf ()
- **basic_streambuf** (const basic_streambuf &)
- void **__safe_gbump** (streamsize __n)
- void **__safe_pbump** (streamsize __n)
- void gbump (int __n)
- virtual void imbue (const locale &__loc)
- basic_streambuf & **operator=** (const basic_streambuf &)
- virtual int_type overflow (int_type __c=traits_type::eof())
- virtual int_type pbackfail (int_type __c=traits_type::eof())
- void pbump (int __n)
- virtual pos_type seekoff (off_type, ios_base::seekdir, ios_base::openmode=ios_base::in|ios_base::out)
- virtual pos_type seekpos (pos_type, ios_base::openmode=ios_base::in|ios_base::out)
- virtual basic_streambuf< char_type, _Traits > * setbuf (char_type *, streamsize)
- void setg (char_type *__gbeg, char_type *__gnext, char_type *__gend)
- void setp (char_type *__pbeg, char_type *__pend)
- virtual streamsize showmanyc ()
- void **swap** (basic_streambuf &__sb)
- virtual int sync ()
- virtual int_type uflow ()
- virtual int_type underflow ()
- virtual streamsize xsgetn (char_type *__s, streamsize __n)
- virtual streamsize xsputn (const char_type *__s, streamsize __n)

- char_type * eback () const
- char_type * gptr () const
- char_type * egptr () const

- char_type * pbase () const
- char_type * pptr () const
- char_type * epptr () const

Protected Attributes

- locale _M_buf_locale
- char_type * _M_in_beg
- char_type * _M_in_cur
- char_type * _M_in_end
- char_type * _M_out_beg
- char_type * _M_out_cur
- char_type * _M_out_end

Friends

- template<bool _IsMove, typename _CharT2>
  __gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, _CharT2 * >::__type **__copy_move_a2** (istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, _CharT2 *)
- streamsize **__copy_streambufs_eof** (basic_streambuf *__sbin, basic_streambuf *__sbout, bool &__ineof)
- void **__istream_extract** (istream &, char *, streamsize)
- template<typename _CharT2, typename _Distance>
  __gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, void >::__type **advance** (istreambuf_iterator< _CharT2 > &, _Distance)
- class **basic_ios< char_type, traits_type >**
- class **basic_istream< char_type, traits_type >**
- class **basic_ostream< char_type, traits_type >**
- template<typename _CharT2>
  __gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, istreambuf_iterator< _CharT2 > >::__type **find** (istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, const _CharT2 &)
- template<typename _CharT2, typename _Traits2, typename _Alloc>
  basic_istream< _CharT2, _Traits2 > & **getline** (basic_istream< _CharT2, _Traits2 > &, basic_string< _CharT2, _Traits2, _Alloc > &, _CharT2)
- class **istreambuf_iterator< char_type, traits_type >**
- template<typename _CharT2, typename _Traits2, typename _Alloc>
  basic_istream< _CharT2, _Traits2 > & **operator>>** (basic_istream< _CharT2, _Traits2 > &, basic_string< _CharT2, _Traits2, _Alloc > &)
- class **ostreambuf_iterator< char_type, traits_type >**

5.231.1 Detailed Description

template<typename _CharT, typename _Traits>
class std::basic_streambuf< _CharT, _Traits >

The actual work of input and output (interface).

Template Parameters

_CharT	Type of character stream.
_Traits	Traits for character type, defaults to char_traits<_CharT>.

This is a base class. Derived stream buffers each control a pair of character sequences: one for input, and one for output.
Section [27.5.1] of the standard describes the requirements and behavior of stream buffer classes. That section (three paragraphs) is reproduced here, for simplicity and accuracy.

1. Stream buffers can impose various constraints on the sequences they control. Some constraints are:
   - The controlled input sequence can be not readable.
   - The controlled output sequence can be not writable.
   - The controlled sequences can be associated with the contents of other representations for character sequences, such as external files.
   - The controlled sequences can support operations *directly* to or from associated sequences.
   - The controlled sequences can impose limitations on how the program can read characters from a sequence, write characters to a sequence, put characters back into an input sequence, or alter the stream position.

2. Each sequence is characterized by three pointers which, if non-null, all point into the same `charT` array object. The array object represents, at any moment, a (sub)sequence of characters from the sequence. Operations performed on a sequence alter the values stored in these pointers, perform reads and writes directly to or from associated sequences, and alter *the stream position* and conversion state as needed to maintain this subsequence relationship. The three pointers are:
   - the *beginning pointer*, or lowest element address in the array (called *xbeg* here);
   - the *next pointer*, or next element address that is a current candidate for reading or writing (called *xnext* here);
   - the *end pointer*, or first element address beyond the end of the array (called *xend* here).
3. The following semantic constraints shall always apply for any set of three pointers for a sequence, using the pointer names given immediately above:
   - If *xnext* is not a null pointer, then *xbeg* and *xend* shall also be non-null pointers into the same `charT` array, as described above; otherwise, *xbeg* and *xend* shall also be null.
   - If *xnext* is not a null pointer and *xnext* < *xend* for an output sequence, then a *write position* is available. In this case, ∗*xnext* shall be assignable as the next element to write (to put, or to store a character value, into the sequence).
   - If *xnext* is not a null pointer and *xbeg* < *xnext* for an input sequence, then a *putback position* is available. In this case, *xnext*[-1] shall have a defined value and is the next (preceding) element to store a character that is put back into the input sequence.
   - If *xnext* is not a null pointer and *xnext*< *xend* for an input sequence, then a *read position* is available. In this case, ∗*xnext* shall have a defined value and is the next element to read (to get, or to obtain a character value, from the sequence).

5.231.2 Member Typedef Documentation

__streambuf_type

```
template<typename _CharT, typename _Traits>
typedef basic_streambuf<char_type, traits_type> std::basic_streambuf< _CharT, _Traits >::__↩
streambuf_type
```

This is a non-standard type.

char_type

```
template<typename _CharT, typename _Traits>
typedef _CharT std::basic_streambuf< _CharT, _Traits >::char_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

int_type

```
template<typename _CharT, typename _Traits>
typedef traits_type::int_type std::basic_streambuf< _CharT, _Traits >::int_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

off_type

```
template<typename _CharT, typename _Traits>
typedef traits_type::off_type std::basic_streambuf< _CharT, _Traits >::off_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

pos_type

```
template<typename _CharT, typename _Traits>
typedef traits_type::pos_type std::basic_streambuf< _CharT, _Traits >::pos_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

traits_type

```
template<typename _CharT, typename _Traits>
typedef _Traits std::basic_streambuf< _CharT, _Traits >::traits_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

5.231.3 Constructor & Destructor Documentation

~basic_streambuf()

```
template<typename _CharT, typename _Traits>
virtual std::basic_streambuf< _CharT, _Traits >::~basic_streambuf ()  [inline], [virtual]
```

Destructor deallocates no buffer space.

basic_streambuf()

```
template<typename _CharT, typename _Traits>
std::basic_streambuf< _CharT, _Traits >::basic_streambuf ()  [inline], [protected]
```

Base constructor.

Only called from derived constructors, and sets up all the buffer data to zero, including the pointers described in the basic_streambuf class description. Note that, as a result,

- the class starts with no read nor write positions available,
- this is not an error

5.231.4 Member Function Documentation

eback()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::eback () const  [inline], [protected]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_stringbuf< _Cha std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_filebuf< _CharT, _Traits >::underflow(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

egptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::egptr () const  [inline], [protected]
```

Access to the get area.
These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Referenced by std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_filebuf< _CharT, _Traits >::underflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), std::wbuffer_convert< _Codecvt, _Elem, _Tr >::underflow(), std::basic_filebuf< _CharT, _Traits >::xsgetn(), and xsgetn().

epptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::epptr () const  [inline], [protected]
```

Access to the put area.
These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_filebuf< _CharT, _Traits >::xsputn(), and xsputn().

gbump()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::gbump (
            int __n)  [inline], [protected]
```

Moving the read position.

Parameters

_↩ _n	The delta by which to move.

This just advances the read position without returning any data.
Referenced by std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

getloc()

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::getloc () const  [inline]
```

Locale access.

Returns

The current locale in effect.

If pubimbue(loc) has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

gptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::gptr () const  [inline], [protected]
```

Access to the get area.
These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_stringbuf< _Cha std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_filebuf< _CharT, _Traits >::underflow(), std::basic_stringbuf< _CharT, _T std::wbuffer_convert< _Codecvt, _Elem, _Tr >::underflow(), std::basic_filebuf< _CharT, _Traits >::xsgetn(), and xsgetn().

imbue()

```
template<typename _CharT, typename _Traits>
virtual void std::basic_streambuf< _CharT, _Traits >::imbue (
            const locale & __loc)  [inline], [protected], [virtual]
```

Changes translations.

Parameters

__loc	A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

Note

Base class version does nothing.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< _CharT, std::char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< wchar_t >, and std::basic_filebuf< wchar_t >.

in_avail()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::in_avail ()  [inline]
```

Looking ahead into the stream.

Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

overflow()

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::overflow (
            int_type __c = traits_type::eof())  [inline], [protected], [virtual]
```

Consumes data from the buffer; writes to the controlled sequence.

Parameters

_↩ _c	An additional character to consume.

Returns

eof() to indicate failure, something else (usually __c, or not_eof())

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if __c is not `eof()`.
For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.
A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

Note

Base class version does nothing, returns eof().

Reimplemented in __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, std::basic_filebuf< char >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< char_type, traits_ std::basic_filebuf< wchar_t >, std::basic_filebuf< wchar_t >, std::basic_stringbuf< _CharT, _Traits, _Alloc >, std::basic_stringbuf< char >, std::basic_stringbuf< char >, std::basic_stringbuf< wchar_t >, std::basic_stringbuf< wchar_t >, and std::wbuffer_convert< _Codecvt, _Elem, _Tr >.
Referenced by xsputn().

pbackfail()

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::pbackfail (
            int_type __c = traits_type::eof())  [inline], [protected], [virtual]
```

Tries to back up the input sequence.

Parameters

_↩ _c	The character to be inserted back into the sequence.

Returns

eof() on failure, *some other value* on success

Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

Note

Base class version does nothing, returns eof().

Reimplemented in __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, std::basic_filebuf< char >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< char_type, traits_ std::basic_filebuf< wchar_t >, std::basic_filebuf< wchar_t >, std::basic_stringbuf< _CharT, _Traits, _Alloc >, std::basic_stringbuf< char >, std::basic_stringbuf< char >, std::basic_stringbuf< wchar_t >, and std::basic_stringbuf< wchar_t >.

pbase()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::pbase () const  [inline], [protected]
```

Access to the put area.
These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, std::basic_filebuf< _CharT, _Traits >::sync(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

pbump()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::pbump (
            int __n)  [inline], [protected]
```

Moving the write position.

Parameters

_↩ _n	The delta by which to move.

This just advances the write position without returning any data.
Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow().

pptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::pptr () const  [inline], [protected]
```

Access to the put area.
These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_filebuf< _CharT, _T std::basic_filebuf< _CharT, _Traits >::xsputn(), and xsputn().

pubimbue()

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::pubimbue (
            const locale & __loc)  [inline]
```

Entry point for imbue().

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls the derived imbue(__loc).

pubseekoff()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekoff (
            off_type __off,
            ios_base::seekdir __way,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [inline]
```

Alters the stream position.

Parameters

__off	Offset.
__way	Value for ios_base::seekdir.
__mode	Value for ios_base::openmode.

Calls virtual seekoff function.

pubseekpos()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekpos (
            pos_type __sp,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [inline]
```

Alters the stream position.

Parameters

__sp	Position
__mode	Value for ios_base::openmode.

Calls virtual seekpos function.

pubsetbuf()

```
template<typename _CharT, typename _Traits>
basic_streambuf * std::basic_streambuf< _CharT, _Traits >::pubsetbuf (
            char_type * __s,
            streamsize __n)  [inline]
```

Entry points for derived buffer functions.
The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

pubsync()

```
template<typename _CharT, typename _Traits>
int std::basic_streambuf< _CharT, _Traits >::pubsync ()  [inline]
```

Calls virtual sync function.
Referenced by std::basic_istream< _CharT, _Traits >::sync().

sbumpc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sbumpc ()  [inline]
```

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.
Referenced by std::basic_istream< _CharT, _Traits >::getline().

seekoff()

```
template<typename _CharT, typename _Traits>
virtual pos_type std::basic_streambuf< _CharT, _Traits >::seekoff (
            off_type ,
            ios_base::seekdir ,
            ios_base::openmode  = ios_base::in | ios_base::out)  [inline], [protected], [virtual]
```

Alters the stream positions.
Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< char_type, traits_
std::basic_filebuf< wchar_t >, std::basic_filebuf< wchar_t >, std::basic_stringbuf< _CharT, _Traits, _Alloc >, std::basic_stringbuf< char >, std::basic_stringbuf< char >, std::basic_stringbuf< wchar_t >, and std::basic_stringbuf< wchar_t >.

seekpos()

```
template<typename _CharT, typename _Traits>
virtual pos_type std::basic_streambuf< _CharT, _Traits >::seekpos (
            pos_type ,
            ios_base::openmode  = ios_base::in | ios_base::out)  [inline], [protected], [virtual]
```

Alters the stream positions.
Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< char_type, traits_
std::basic_filebuf< wchar_t >, std::basic_filebuf< wchar_t >, std::basic_stringbuf< _CharT, _Traits, _Alloc >, std::basic_stringbuf< char >, std::basic_stringbuf< char >, std::basic_stringbuf< wchar_t >, and std::basic_stringbuf< wchar_t >.

setbuf()

```
template<typename _CharT, typename _Traits>
virtual basic_streambuf< char_type, _Traits > * std::basic_streambuf< _CharT, _Traits >::setbuf
(
            char_type * ,
            streamsize )  [inline], [protected], [virtual]
```

Manipulates the buffer.
Each derived class provides its own appropriate behavior. See the next-to-last paragraph of https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html#io.streambuf.buffering for more on this function.

Note

Base class version does nothing, returns `this`.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< _CharT, std::char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< wchar_t >, std::basic_stringbuf< _CharT, _Traits, _Alloc >, std::basic_stringbuf< char >, and std::basic_stringbuf< char >.

setg()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setg (
            char_type * __gbeg,
            char_type * __gnext,
            char_type * __gend)  [inline], [protected]
```

Setting the three read area pointers.

Parameters

__gbeg	A pointer.
__gnext	A pointer.
__gend	A pointer.

Postcondition

__gbeg == `eback()`, *__gnext* == `gptr()`, and *__gend* == `egptr()`

Referenced by std::wbuffer_convert< _Codecvt, _Elem, _Tr >::wbuffer_convert(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflo
std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(),
and std::basic_filebuf< _CharT, _Traits >::xsgetn().

setp()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setp (
            char_type * __pbeg,
            char_type * __pend)  [inline], [protected]
```

Setting the three write area pointers.

Parameters

__pbeg	A pointer.
__pend	A pointer.

Postcondition

__pbeg == `pbase()`, *__pbeg* == `pptr()`, and *__pend* == `epptr()`

Referenced by std::wbuffer_convert< _Codecvt, _Elem, _Tr >::wbuffer_convert().

sgetc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sgetc ()  [inline]
```

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.
Referenced by std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), and std::basic_istream< _CharT, _Traits >::ignore().

sgetn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sgetn (
            char_type * __s,
            streamsize __n)  [inline]
```

Entry point for xsgetn.

Parameters

_↩ _s	A buffer area.
_↩ _n	A count.

Returns xsgetn(__s,__n). The effect is to fill __s[0] through __s[__n-1] with characters from the input sequence, if possible.

showmanyc()

```
template<typename _CharT, typename _Traits>
virtual streamsize std::basic_streambuf< _CharT, _Traits >::showmanyc ()  [inline], [protected],
[virtual]
```

Investigating the data available.

Returns

An estimate of the number of characters available in the input sequence, or -1.

If it returns a positive value, then successive calls to `underflow()` *will not return* `traits::eof()` *until at least that number of characters have been supplied. If* `showmanyc()` *returns -1, then calls to* `underflow()` *or* `uflow()` *will fail.* [27.5.2.4.3]/1

Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to underflow or uflow] will not return* `eof()` *but that they will return immediately.*

The standard adds that *the morphemes of* `showmanyc` *are* ***es-how-many-see****, not* ***show-manic****.*

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< _CharT, std::char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< wchar_t >, std::basic_filebuf< wchar_t >, std::basic_stringbuf< _CharT, _Traits, _Alloc >, std::basic_stringbuf< char >, std::basic_stringbuf< char std::basic_stringbuf< wchar_t >, and std::basic_stringbuf< wchar_t >.

snextc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::snextc ()  [inline]
```

Getting the next character.

Returns

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.
Referenced by std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< char_type, traits_type >::ignore(), std::basic_istream< char_type, traits_type >::operator>>(), and std::basic_istream< char_type, traits_type >::putback().

sputbackc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputbackc (
            char_type __c)  [inline]
```

Pushing characters back into the input stream.

Parameters

__c	The character to push back.

Returns

The previous character, if possible.

Similar to sungetc(), but *__c* is pushed onto the stream instead of *the previous character.* If successful, the next character fetched from the input stream will be *__c*.
Referenced by std::basic_istream< _CharT, _Traits >::putback().

sputc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputc (
            char_type __c)  [inline]
```

Entry point for all single-character output functions.

Parameters

_↩ _c	A character to output.

Returns

__c, if possible.

One of two public output functions.
If a write position is available for the output sequence (i.e., the buffer is not full), stores __c in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(↩ __c)`.
Referenced by std::wbuffer_convert< _Codecvt, _Elem, _Tr >::overflow().

sputn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sputn (
            const char_type * __s,
            streamsize __n)  [inline]
```

Entry point for all single-character output functions.

Parameters

_↩ _s	A buffer read area.
_↩ _n	A count.

One of two public output functions.
Returns xsputn(__s,__n). The effect is to write __s[0] through __s[__n-1] to the output sequence, if possible.

sungetc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sungetc ()  [inline]
```

Moving backwards in the input stream.

Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns pbackfail(). The effect is to *unget* the last character *gotten*.
Referenced by std::basic_istream< char_type, traits_type >::sentry::sentry().

sync()

```
template<typename _CharT, typename _Traits>
virtual int std::basic_streambuf< _CharT, _Traits >::sync ()  [inline], [protected], [virtual]
```

Synchronizes the buffer arrays with the controlled sequences.

Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

Note

Base class version does nothing, returns zero.

Reimplemented in __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, std::basic_filebuf< _CharT, std::char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< wchar_t >, std::basic_filebuf< wchar_t >, and std::wbuffer_convert< _Codecvt, _Elem, _Tr >.

uflow()

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::uflow ()  [inline], [protected], [virtual]
```

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.
Reimplemented in __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >.
Referenced by xsgetn().

underflow()

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::underflow ()  [inline], [protected],
[virtual]
```

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.
For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.
A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html

Note

Base class version does nothing, returns eof().

Reimplemented in __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, std::basic_filebuf< _CharT, std::char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< wchar_t >, std::basic_filebuf< wchar_t >, std::basic_stringbuf< _CharT, _Traits, _Alloc >, std::basic_stringbuf< char >, std::basic_stringbuf< char std::basic_stringbuf< wchar_t >, std::basic_stringbuf< wchar_t >, and std::wbuffer_convert< _Codecvt, _Elem, _Tr >.

xsgetn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::xsgetn (
            char_type * __s,
            streamsize __n)  [protected], [virtual]
```

Multiple character extraction.

Parameters

_↩ _s	A buffer area.
_↩ _n	Maximum number of characters to assign.

Returns

The number of characters assigned.

Fills __s[0] through __s[__n-1] with characters from the input sequence, as if by `sbumpc()`. Stops when either __n characters have been copied, or when `traits::eof()` would be copied.
It is expected that derived classes provide a more efficient implementation by overriding this definition.
Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< _CharT, std::char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< char_type, traits_type >, and std::basic_filebuf< wchar_t >.
References egptr(), gptr(), std::min(), and uflow().

xsputn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::xsputn (
            const char_type * __s,
            streamsize __n)  [protected], [virtual]
```

Multiple character insertion.

Parameters

_↩ _s	A buffer area.
_↩ _n	Maximum number of characters to write.

Returns

The number of characters written.

Writes __s[0] through __s[__n-1] to the output sequence, as if by `sputc()`. Stops when either *n* characters have been copied, or when `sputc()` would return `traits::eof()`.
It is expected that derived classes provide a more efficient implementation by overriding this definition.
Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< _CharT, std::char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< char_type, traits_type >, and std::basic_filebuf< wchar_t >.
References epptr(), std::min(), overflow(), and pptr().

5.231.5 Member Data Documentation

_M_buf_locale

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::_M_buf_locale  [protected]
```

Current locale setting.
Referenced by std::basic_filebuf< _CharT, _Traits >::basic_filebuf().

_M_in_beg

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_beg  [protected]
```
Start of get area.

_M_in_cur

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_cur  [protected]
```
Current read area.

_M_in_end

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_end  [protected]
```
End of get area.

_M_out_beg

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_beg  [protected]
```
Start of put area.

_M_out_cur

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_cur  [protected]
```
Current put area.

_M_out_end

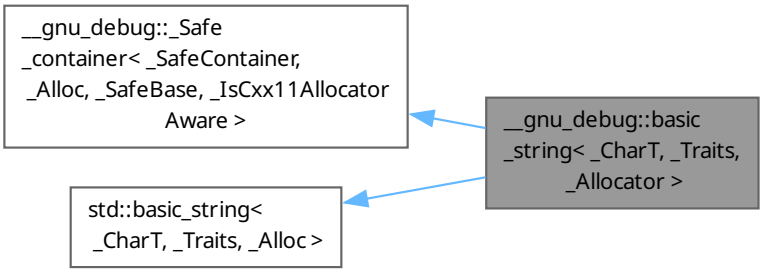
```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_end  [protected]
```
End of put area.
The documentation for this class was generated from the following files:

- iosfwd
- streambuf
- streambuf.tcc

5.232 __gnu_debug::basic_string< _CharT, _Traits, _Allocator > Class Template Reference

```
#include <string>
```

Inheritance diagram for __gnu_debug::basic_string< _CharT, _Traits, _Allocator >:



Public Types

- typedef _Allocator **allocator_type**
- typedef __gnu_debug::_Safe_iterator< typename _Base::const_iterator, basic_string > **const_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**
- typedef std::reverse_iterator< const_iterator > **const_reverse_iterator**
- typedef _Base::difference_type **difference_type**
- typedef __gnu_debug::_Safe_iterator< typename _Base::iterator, basic_string > **iterator**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef std::reverse_iterator< iterator > **reverse_iterator**
- typedef _Base::size_type **size_type**
- typedef _Traits **traits_type**
- typedef _Traits::char_type **value_type**

Public Member Functions

- **basic_string** (_Base &&__base) noexcept
- template<typename _InputIterator>
  basic_string (_InputIterator __begin, _InputIterator __end, const _Allocator &__a=_Allocator())
- **basic_string** (basic_string &&)=default
- **basic_string** (basic_string &&__s, const _Allocator &__a) noexcept(std::is_nothrow_constructible< _Base, _Base, const _Allocator & >::value)
- **basic_string** (const _Allocator &__a) noexcept
- **basic_string** (const _Base &__base)
- **basic_string** (const _CharT *__s, const _Allocator &__a=_Allocator())
- **basic_string** (const _CharT *__s, size_type __n, const _Allocator &__a=_Allocator())
- **basic_string** (const basic_string &)=default
- **basic_string** (const basic_string &__s, const _Allocator &__a)
- **basic_string** (const basic_string &__str, size_type __pos, size_type __n=_Base::npos, const _Allocator &__a=_Allocator())
- **basic_string** (size_type __n, _CharT __c, const _Allocator &__a=_Allocator())
- **basic_string** (std::initializer_list< _CharT > __l, const _Allocator &__a=_Allocator())

- template<typename _Operation>
  constexpr void __resize_and_overwrite (size_type __n, _Operation __op)
- template<typename _Operation>
  void __resize_and_overwrite (size_type __n, _Operation __op)
- const _Base & **_M_base** () const noexcept
- _Base & **_M_base** () noexcept
- template<typename _InputIterator>
  basic_string< _CharT, _Traits, _Alloc > & **_M_replace_dispatch** (iterator __i1, iterator __i2, _InputIterator __k1, _InputIterator __k2, __false_type)
- template<typename _InIterator>
  _CharT * **_S_construct** (_InIterator __beg, _InIterator __end, const _Alloc &__a, forward_iterator_tag)
- template<typename _InputIterator>
  basic_string & **append** (_InputIterator __first, _InputIterator __last)
- basic_string & **append** (const _CharT *__s)
- basic_string & **append** (const _CharT *__s, size_type __n)
- basic_string & **append** (const basic_string &__str)
- basic_string & **append** (const basic_string &__str, size_type __pos, size_type __n)
- basic_string & **append** (size_type __n, _CharT __c)
- constexpr basic_string & append (const _CharT *__s, size_type __n)
- constexpr basic_string & append (const basic_string &__str)
- constexpr basic_string & append (const basic_string &__str, size_type __pos, size_type __n=npos)
- constexpr basic_string & append (initializer_list< _CharT > __l)
- constexpr basic_string & append (size_type __n, _CharT __c)
- template<typename _InputIterator>
  basic_string & **assign** (_InputIterator __first, _InputIterator __last)
- basic_string & **assign** (basic_string &&__x) noexcept(noexcept(std::declval< _Base & >().assign(std::move(_↩
  _x))))
- basic_string & **assign** (const _CharT *__s)
- basic_string & **assign** (const _CharT *__s, size_type __n)
- basic_string & **assign** (const basic_string &__str, size_type __pos, size_type __n)
- basic_string & **assign** (const basic_string &__x)
- basic_string & **assign** (size_type __n, _CharT __c)
- basic_string & **assign** (std::initializer_list< _CharT > __l)
- constexpr basic_string & assign (basic_string &&__str) noexcept(_Alloc_traits::_S_nothrow_move())
- constexpr basic_string & assign (const _CharT *__s, size_type __n)
- constexpr basic_string & assign (const basic_string &__str)
- constexpr basic_string & assign (const basic_string &__str, size_type __pos, size_type __n=npos)
- constexpr basic_string & assign (size_type __n, _CharT __c)
- constexpr reference at (size_type __n)
- reference at (size_type __n)
- constexpr const_reference at (size_type __n) const
- const_reference at (size_type __n) const
- constexpr reference at (size_type __n)
- constexpr const_reference at (size_type __n) const
- constexpr const_reference back () const noexcept
- constexpr reference back () noexcept
- iterator **begin** ()
- const_iterator **begin** () const noexcept
- const _CharT * **c_str** () const noexcept
- constexpr size_type capacity () const noexcept
- const_iterator **cbegin** () const noexcept

- const_iterator **cend** () const noexcept
- void **clear** ()
- constexpr int **compare** (const _CharT *__s) const noexcept
- constexpr int compare (const basic_string &__str) const
- constexpr int compare (size_type __pos, size_type __n, const basic_string &__str) const
- int compare (size_type __pos, size_type __n, const basic_string &__str) const
- constexpr int compare (size_type __pos, size_type __n1, const _CharT *__s) const
- constexpr int compare (size_type __pos, size_type __n1, const _CharT *__s, size_type __n2) const
- int compare (size_type __pos, size_type __n1, const _CharT *__s, size_type __n2) const
- constexpr int **compare** (size_type __pos1, size_type __n1, const _CharT *__s) const
- constexpr int **compare** (size_type __pos1, size_type __n1, const _CharT *__s, size_type __n2) const
- constexpr int compare (size_type __pos1, size_type __n1, const basic_string &__str, size_type __pos2, size_type __n2=npos) const
- int compare (size_type __pos1, size_type __n1, const basic_string &__str, size_type __pos2, size_type __n2=npos) const
- constexpr int compare (size_type __pos, size_type __n, const basic_string &__str) const
- constexpr int compare (size_type __pos, size_type __n1, const _CharT *__s) const
- constexpr int compare (size_type __pos, size_type __n1, const _CharT *__s, size_type __n2) const
- constexpr int compare (size_type __pos1, size_type __n1, const basic_string &__str, size_type __pos2, size_type __n2=npos) const
- size_type **copy** (_CharT *__s, size_type __n, size_type __pos=0) const
- constexpr size_type copy (_CharT *__s, size_type __n, size_type __pos=0) const
- const_reverse_iterator **crbegin** () const noexcept
- const_reverse_iterator **crend** () const noexcept
- const _CharT * **data** () const noexcept
- constexpr _CharT * data () noexcept
- constexpr bool empty () const noexcept
- iterator **end** ()
- const_iterator **end** () const noexcept
- constexpr bool **ends_with** (_CharT __x) const noexcept
- bool **ends_with** (_CharT __x) const noexcept
- constexpr bool **ends_with** (basic_string_view< _CharT, _Traits > __x) const noexcept
- bool **ends_with** (basic_string_view< _CharT, _Traits > __x) const noexcept
- constexpr bool **ends_with** (const _CharT *__x) const noexcept
- bool **ends_with** (const _CharT *__x) const noexcept
- iterator **erase** (__const_iterator __first, __const_iterator __last)
- iterator **erase** (__const_iterator __position)
- basic_string & **erase** (size_type __pos=0, size_type __n=_Base::npos)
- constexpr iterator erase (__const_iterator __first, __const_iterator __last)
- constexpr iterator erase (__const_iterator __position)
- iterator erase (iterator __first, iterator __last)
- iterator erase (iterator __position)
- constexpr basic_string & erase (size_type __pos=0, size_type __n=npos)
- constexpr size_type find (_CharT __c, size_type __pos=0) const noexcept
- size_type find (_CharT __c, size_type __pos=0) const noexcept
- constexpr size_type find (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- size_type find (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type **find** (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type find (const _CharT *__s, size_type __pos=0) const noexcept
- constexpr size_type **find** (const _CharT *__s, size_type __pos=0) const noexcept
- constexpr size_type find (const basic_string &__str, size_type __pos=0) const noexcept

- size_type find (const basic_string &__str, size_type __pos=0) const noexcept
- constexpr size_type find (_CharT __c, size_type __pos=0) const noexcept
- constexpr size_type find (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type find (const _CharT *__s, size_type __pos=0) const noexcept
- constexpr size_type find (const basic_string &__str, size_type __pos=0) const noexcept
- constexpr size_type find_first_not_of (_CharT __c, size_type __pos=0) const noexcept
- size_type find_first_not_of (_CharT __c, size_type __pos=0) const noexcept
- constexpr size_type find_first_not_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- size_type find_first_not_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type **find_first_not_of** (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type find_first_not_of (const _CharT *__s, size_type __pos=0) const noexcept
- constexpr size_type **find_first_not_of** (const _CharT *__s, size_type __pos=0) const noexcept
- constexpr size_type find_first_not_of (const basic_string &__str, size_type __pos=0) const noexcept
- size_type find_first_not_of (const basic_string &__str, size_type __pos=0) const noexcept
- constexpr size_type find_first_not_of (_CharT __c, size_type __pos=0) const noexcept
- constexpr size_type find_first_not_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type find_first_not_of (const _CharT *__s, size_type __pos=0) const noexcept
- constexpr size_type find_first_not_of (const basic_string &__str, size_type __pos=0) const noexcept
- constexpr size_type find_first_of (_CharT __c, size_type __pos=0) const noexcept
- size_type find_first_of (_CharT __c, size_type __pos=0) const noexcept
- constexpr size_type find_first_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- size_type find_first_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type **find_first_of** (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type find_first_of (const _CharT *__s, size_type __pos=0) const noexcept
- constexpr size_type **find_first_of** (const _CharT *__s, size_type __pos=0) const noexcept
- constexpr size_type find_first_of (const basic_string &__str, size_type __pos=0) const noexcept
- size_type find_first_of (const basic_string &__str, size_type __pos=0) const noexcept
- constexpr size_type find_first_of (_CharT __c, size_type __pos=0) const noexcept
- constexpr size_type find_first_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type find_first_of (const _CharT *__s, size_type __pos=0) const noexcept
- constexpr size_type find_first_of (const basic_string &__str, size_type __pos=0) const noexcept
- constexpr size_type find_last_not_of (_CharT __c, size_type __pos=npos) const noexcept
- size_type find_last_not_of (_CharT __c, size_type __pos=npos) const noexcept
- constexpr size_type find_last_not_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- size_type find_last_not_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type **find_last_not_of** (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type **find_last_not_of** (const _CharT *__s, size_type __pos=_Base::npos) const noexcept
- constexpr size_type find_last_not_of (const _CharT *__s, size_type __pos=npos) const noexcept
- constexpr size_type find_last_not_of (const basic_string &__str, size_type __pos=npos) const noexcept
- size_type find_last_not_of (const basic_string &__str, size_type __pos=npos) const noexcept
- constexpr size_type find_last_not_of (_CharT __c, size_type __pos=npos) const noexcept
- constexpr size_type find_last_not_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type find_last_not_of (const _CharT *__s, size_type __pos=npos) const noexcept
- constexpr size_type find_last_not_of (const basic_string &__str, size_type __pos=npos) const noexcept
- constexpr size_type find_last_of (_CharT __c, size_type __pos=npos) const noexcept
- size_type find_last_of (_CharT __c, size_type __pos=npos) const noexcept
- constexpr size_type find_last_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- size_type find_last_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type **find_last_of** (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type **find_last_of** (const _CharT *__s, size_type __pos=_Base::npos) const noexcept
- constexpr size_type find_last_of (const _CharT *__s, size_type __pos=npos) const noexcept

- constexpr size_type find_last_of (const basic_string &__str, size_type __pos=npos) const noexcept
- size_type find_last_of (const basic_string &__str, size_type __pos=npos) const noexcept
- constexpr size_type find_last_of (_CharT __c, size_type __pos=npos) const noexcept
- constexpr size_type find_last_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type find_last_of (const _CharT *__s, size_type __pos=npos) const noexcept
- constexpr size_type find_last_of (const basic_string &__str, size_type __pos=npos) const noexcept
- constexpr const_reference front () const noexcept
- constexpr reference front () noexcept
- constexpr allocator_type get_allocator () const noexcept
- iterator **insert** (__const_iterator __p, _CharT __c)
- template<typename _InputIterator>
  iterator **insert** (__const_iterator __p, _InputIterator __first, _InputIterator __last)
- iterator **insert** (const_iterator __p, size_type __n, _CharT __c)
- iterator **insert** (const_iterator __p, std::initializer_list< _CharT > __l)
- basic_string & **insert** (size_type __pos, const _CharT *__s)
- basic_string & **insert** (size_type __pos, const _CharT *__s, size_type __n)
- basic_string & **insert** (size_type __pos, size_type __n, _CharT __c)
- basic_string & **insert** (size_type __pos1, const basic_string &__str)
- basic_string & **insert** (size_type __pos1, const basic_string &__str, size_type __pos2, size_type __n)
- constexpr iterator insert (__const_iterator __p, _CharT __c)
- template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  constexpr iterator insert (const_iterator __p, _InputIterator __beg, _InputIterator __end)
- constexpr iterator insert (const_iterator __p, initializer_list< _CharT > __l)
- constexpr iterator insert (const_iterator __p, size_type __n, _CharT __c)
- iterator insert (iterator __p, _CharT __c)
- template<class _InputIterator>
  void insert (iterator __p, _InputIterator __beg, _InputIterator __end)
- void insert (iterator __p, initializer_list< _CharT > __l)
- void insert (iterator __p, size_type __n, _CharT __c)
- constexpr basic_string & insert (size_type __pos, const _CharT *__s)
- constexpr basic_string & insert (size_type __pos, const _CharT *__s, size_type __n)
- constexpr basic_string & insert (size_type __pos, size_type __n, _CharT __c)
- constexpr basic_string & insert (size_type __pos1, const basic_string &__str)
- constexpr basic_string & insert (size_type __pos1, const basic_string &__str, size_type __pos2, size_type __n=npos)
- constexpr size_type length () const noexcept
- constexpr size_type max_size () const noexcept
- basic_string & **operator+=** (_CharT __c)
- basic_string & **operator+=** (const _CharT *__s)
- basic_string & **operator+=** (const basic_string &__str)
- basic_string & **operator+=** (std::initializer_list< _CharT > __l)
- constexpr basic_string & operator+= (const basic_string &__str)
- basic_string & **operator=** (_CharT __c)
- basic_string & **operator=** (basic_string &&)=default
- basic_string & **operator=** (const _CharT *__s)
- basic_string & **operator=** (const basic_string &)=default
- basic_string & **operator=** (std::initializer_list< _CharT > __l)
- reference **operator[ ]** (size_type __pos)
- const_reference **operator[ ]** (size_type __pos) const noexcept
- constexpr reference operator[ ] (size_type __pos)
- constexpr const_reference operator[ ] (size_type __pos) const noexcept

- void **pop_back** ()
- void **push_back** (_CharT __c)
- reverse_iterator **rbegin** ()
- const_reverse_iterator **rbegin** () const noexcept
- reverse_iterator **rend** ()
- const_reverse_iterator **rend** () const noexcept
- template<typename _InputIterator>
  basic_string & **replace** (__const_iterator __i1, __const_iterator __i2, _InputIterator __j1, _InputIterator __j2)
- basic_string & **replace** (__const_iterator __i1, __const_iterator __i2, const _CharT *__s)
- basic_string & **replace** (__const_iterator __i1, __const_iterator __i2, const _CharT *__s, size_type __n)
- basic_string & **replace** (__const_iterator __i1, __const_iterator __i2, const basic_string &__str)
- basic_string & **replace** (__const_iterator __i1, __const_iterator __i2, size_type __n, _CharT __c)
- basic_string & **replace** (__const_iterator __i1, __const_iterator __i2, std::initializer_list< _CharT > __l)
- basic_string & **replace** (size_type __pos, size_type __n1, const _CharT *__s)
- basic_string & **replace** (size_type __pos, size_type __n1, const _CharT *__s, size_type __n2)
- basic_string & **replace** (size_type __pos, size_type __n1, size_type __n2, _CharT __c)
- basic_string & **replace** (size_type __pos1, size_type __n1, const basic_string &__str)
- basic_string & **replace** (size_type __pos1, size_type __n1, const basic_string &__str, size_type __pos2, size_type __n2)
- constexpr basic_string & **replace** (__const_iterator __i1, __const_iterator __i2, _CharT *__k1, _CharT *__k2)
- constexpr basic_string & **replace** (__const_iterator __i1, __const_iterator __i2, const _CharT *__k1, const _CharT *__k2)
- constexpr basic_string & replace (__const_iterator __i1, __const_iterator __i2, const _CharT *__s)
- constexpr basic_string & replace (__const_iterator __i1, __const_iterator __i2, const _CharT *__s, size_type __n)
- constexpr basic_string & replace (__const_iterator __i1, __const_iterator __i2, const basic_string &__str)
- constexpr basic_string & **replace** (__const_iterator __i1, __const_iterator __i2, const_iterator __k1, const_iterator __k2)
- constexpr basic_string & **replace** (__const_iterator __i1, __const_iterator __i2, iterator __k1, iterator __k2)
- constexpr basic_string & replace (__const_iterator __i1, __const_iterator __i2, size_type __n, _CharT __c)
- template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  constexpr basic_string & replace (const_iterator __i1, const_iterator __i2, _InputIterator __k1, _InputIterator __k2)
- constexpr basic_string & replace (const_iterator __i1, const_iterator __i2, initializer_list< _CharT > __l)
- basic_string & **replace** (iterator __i1, iterator __i2, _CharT *__k1, _CharT *__k2)
- template<class _InputIterator>
  basic_string & replace (iterator __i1, iterator __i2, _InputIterator __k1, _InputIterator __k2)
- basic_string & **replace** (iterator __i1, iterator __i2, const _CharT *__k1, const _CharT *__k2)
- basic_string & replace (iterator __i1, iterator __i2, const _CharT *__s)
- basic_string & replace (iterator __i1, iterator __i2, const _CharT *__s, size_type __n)
- basic_string & replace (iterator __i1, iterator __i2, const basic_string &__str)
- basic_string & **replace** (iterator __i1, iterator __i2, const_iterator __k1, const_iterator __k2)
- basic_string & replace (iterator __i1, iterator __i2, initializer_list< _CharT > __l)
- basic_string & **replace** (iterator __i1, iterator __i2, iterator __k1, iterator __k2)
- basic_string & replace (iterator __i1, iterator __i2, size_type __n, _CharT __c)
- constexpr basic_string & replace (size_type __pos, size_type __n, const basic_string &__str)
- constexpr basic_string & replace (size_type __pos, size_type __n1, const _CharT *__s)
- constexpr basic_string & replace (size_type __pos, size_type __n1, const _CharT *__s, size_type __n2)
- constexpr basic_string & replace (size_type __pos, size_type __n1, size_type __n2, _CharT __c)
- constexpr basic_string & replace (size_type __pos1, size_type __n1, const basic_string &__str, size_type __pos2, size_type __n2=npos)
- constexpr void reserve ()
- constexpr void reserve (size_type __res_arg)

- void reserve (size_type __res_arg)
- constexpr void reserve (size_type __res_arg)
- void **resize** (size_type __n)
- void **resize** (size_type __n, _CharT __c)
- constexpr void resize (size_type __n)
- constexpr void resize (size_type __n, _CharT __c)
- constexpr size_type rfind (_CharT __c, size_type __pos=npos) const noexcept
- size_type rfind (_CharT __c, size_type __pos=npos) const noexcept
- constexpr size_type **rfind** (const _CharT *__s, size_type __pos, size_type __n) const
- constexpr size_type rfind (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- size_type rfind (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type **rfind** (const _CharT *__s, size_type __pos=_Base::npos) const
- constexpr size_type rfind (const _CharT *__s, size_type __pos=npos) const
- constexpr size_type rfind (const basic_string &__str, size_type __pos=npos) const noexcept
- size_type rfind (const basic_string &__str, size_type __pos=npos) const noexcept
- constexpr size_type rfind (_CharT __c, size_type __pos=npos) const noexcept
- constexpr size_type rfind (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type rfind (const _CharT *__s, size_type __pos=npos) const
- constexpr size_type rfind (const basic_string &__str, size_type __pos=npos) const noexcept
- void **shrink_to_fit** () noexcept
- constexpr size_type size () const noexcept
- constexpr bool **starts_with** (_CharT __x) const noexcept
- bool **starts_with** (_CharT __x) const noexcept
- constexpr bool **starts_with** (basic_string_view< _CharT, _Traits > __x) const noexcept
- bool **starts_with** (basic_string_view< _CharT, _Traits > __x) const noexcept
- constexpr bool **starts_with** (const _CharT *__x) const noexcept
- bool **starts_with** (const _CharT *__x) const noexcept
- basic_string **substr** (size_type __pos=0, size_type __n=_Base::npos) const
- constexpr basic_string substr (size_type __pos=0, size_type __n=npos) const
- void **swap** (basic_string &__x) noexcept(/*conditional */)
- constexpr void swap (basic_string &__s) noexcept

Static Public Attributes

- static const size_type npos

Protected Member Functions

- constexpr void **_M_swap** (const _Safe_container &__x) const noexcept

Friends

- template<typename _ItT, typename _SeqT, typename _CatT>
  class **::__gnu_debug::_Safe_iterator**

5.232.1 Detailed Description

template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>>
class __gnu_debug::basic_string< _CharT, _Traits, _Allocator >

Class std::basic_string with safety/checking/debug instrumentation.

5.232.2 Member Function Documentation

__resize_and_overwrite() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<typename _Operation>
void std::basic_string< _CharT, _Traits, _Alloc >::__resize_and_overwrite (
            size_type __n,
            _Operation __op)  [constexpr], [inherited]
```

Non-standard version of resize_and_overwrite for C++11 and above.

__resize_and_overwrite() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<typename _Operation>
void std::basic_string< _CharT, _Traits, _Alloc >::__resize_and_overwrite (
            size_type __n,
            _Operation __op)  [inherited]
```

Non-standard version of resize_and_overwrite for C++11 and above.

append() [1/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::append (
            const _CharT * __s,
            size_type __n)  [inline], [constexpr], [inherited]
```

Append a C substring.

Parameters

__s	The C string to append.
__n	The number of characters to append.

Returns

Reference to this string.

append() [2/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::append (
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline], [constexpr], [inherited]
```

Append a string to this string.

Parameters

__str	The string to append.

Returns

Reference to this string.

Referenced by std::basic_string< _CharT >::append(), std::basic_string< _CharT >::append(), std::basic_string< _CharT >::append(), std::basic_string< _CharT >::append(), std::__cxx11::collate< _CharT >::do_transform(), std::basic_string< _CharT >::operator+=(), std::basic_string< _CharT >::operator+=(), std::basic_string< _CharT >::operator+=(), std::operator>>(), and resize().

append() [3/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::append (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos,
            size_type __n = npos)  [inline], [constexpr], [inherited]
```

Append a substring.

Parameters

__str	The string to append.
__pos	Index of the first character of str to append.
__n	The number of characters to append.

Returns

Reference to this string.

Exceptions

std::out_of_range	if *__pos* is not a valid index.

This function appends *__n* characters from *__str* starting at *__pos* to this string. If *__n* is is larger than the number of available characters in *__str*, the remainder of *__str* is appended.

append() [4/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::append (
            initializer_list< _CharT > __l)  [inline], [constexpr], [inherited]
```

Append an initializer_list of characters.

Parameters

__l	The initializer_list of characters to append.

Returns

Reference to this string.

append() [5/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::append (
            size_type __n,
            _CharT __c)  [inline], [constexpr], [inherited]
```

Append multiple characters.

Parameters

_↩ _n	The number of characters to append.
_↩ _c	The character to use.

Returns

Reference to this string.

Appends __n copies of __c to this string.

assign() [1/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
            basic_string< _CharT, _Traits, _Alloc > && __str)  [inline], [constexpr], [noexcept],
[inherited]
```

Set value to contents of another string.

Parameters

__str	Source string to use.

Returns

Reference to this string.

This function sets this string to the exact contents of __str. __str is a valid, but unspecified string.

assign() [2/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::assign (
            const _CharT * __s,
            size_type __n)  [inline], [constexpr], [inherited]
```

Set value to a C substring.

Parameters

_↩ _s	The C string to use.
_↩ _n	Number of characters to use.

Returns

Reference to this string.

This function sets the value of this string to the first __n characters of __s. If __n is is larger than the number of available characters in __s, the remainder of __s is used.

assign() [3/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::assign (
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline], [constexpr], [inherited]
```

Set value to contents of another string.

Parameters

__str	Source string to use.

Returns

Reference to this string.

Referenced by std::basic_string< _CharT >::assign(), std::basic_string< _CharT >::assign(), std::basic_string< _CharT >::assign(), __gnu_debug::basic_string< char >::compare(), std::basic_string< _CharT >::operator=(), std::basic_string< _CharT >::operator=(), std::basic_string< _CharT >::operator=(), std::basic_string< _CharT >::operator=(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >

assign() [4/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos,
            size_type __n = npos)  [inline], [constexpr], [inherited]
```

Set value to a substring of a string.

Parameters

__str	The string to use.
__pos	Index of the first character of str.
__n	Number of characters to use.

Returns

Reference to this string.

Exceptions

std::out_of_range	if *pos* is not a valid index.

This function sets this string to the substring of __str consisting of __n characters at __pos. If __n is is larger than the number of available characters in __str, the remainder of __str is used.

assign() [5/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
            size_type __n,
            _CharT __c)  [inline], [constexpr], [inherited]
```

Set value to multiple characters.

Parameters

_↩ _n	Length of the resulting string.
_↩ _c	The character to use.

Returns

Reference to this string.

This function sets the value of this string to __n copies of character __c.

at() [1/6]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
reference std::basic_string< _CharT, _Traits, _Allocator >::at (
            size_type __n)  [inline], [nodiscard], [constexpr]
```

Provides access to the data contained in the string.

Parameters

_↩ _n	The index of the character to access.

Returns

Read/write reference to the character.

Exceptions

std::out_of_range	If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws out_of_range if the check fails.

at() [2/6]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
reference std::basic_string< _CharT, _Traits, _Allocator >::at (
            size_type __n)  [inline]
```

Provides access to the data contained in the string.

Parameters

_↩ _n	The index of the character to access.

Returns

Read/write reference to the character.

Exceptions

std::out_of_range	If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws out_of_range if the check fails. Success results in unsharing the string.

at() [3/6]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
const_reference std::basic_string< _CharT, _Traits, _Allocator >::at (
            size_type __n) const  [inline], [nodiscard], [constexpr]
```

Provides access to the data contained in the string.

Parameters

_↩ _n	The index of the character to access.

Returns

Read-only (const) reference to the character.

Exceptions

std::out_of_range	If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws out_of_range if the check fails.

at() [4/6]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
const_reference std::basic_string< _CharT, _Traits, _Allocator >::at (
            size_type __n) const  [inline]
```

Provides access to the data contained in the string.

Parameters

_↩ _n	The index of the character to access.

Returns

Read-only (const) reference to the character.

Exceptions

std::out_of_range	If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws out_of_range if the check fails.

at() [5/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string< _CharT, _Traits, _Alloc >::at (
            size_type __n)  [inline], [nodiscard], [constexpr], [inherited]
```

Provides access to the data contained in the string.

Parameters

_↩ _n	The index of the character to access.

Returns

Read/write reference to the character.

Exceptions

std::out_of_range	If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws out_of_range if the check fails.

at() [6/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string< _CharT, _Traits, _Alloc >::at (
            size_type __n) const  [inline], [nodiscard], [constexpr], [inherited]
```

Provides access to the data contained in the string.

Parameters

_↩ _n	The index of the character to access.

Returns

Read-only (const) reference to the character.

Exceptions

std::out_of_range	If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws out_of_range if the check fails.

back() [1/2]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
const_reference std::basic_string< _CharT, _Traits, _Allocator >::back () const  [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns a read-only (constant) reference to the data at the last element of the string.

back() [2/2]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
reference std::basic_string< _CharT, _Traits, _Allocator >::back ()  [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read/write reference to the data at the last element of the string.

capacity()

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::capacity () const  [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns the total number of characters that the string can hold before needing to allocate more memory.

compare() [1/12]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
int std::basic_string< _CharT, _Traits, _Allocator >::compare (
            const basic_string< _CharT, _Traits, _Allocator > & __str) const  [inline], [nodiscard],
[constexpr]
```

Compare to a string.

Parameters

__str	String to compare against.

Returns

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before *__str*, 0 if their values are equivalent, or > 0 if this string is ordered after *__str*. Determines the effective length rlen of the strings to compare as the smallest of size() and str.size(). The function then compares the two strings by calling traits::compare(data(), str.data(),rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

compare() [2/12]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
int std::basic_string< _CharT, _Traits, _Allocator >::compare (
            size_type __pos,
            size_type __n,
            const basic_string< _CharT, _Traits, _Allocator > & __str) const  [inline], [nodiscard],
[constexpr]
```

Compare substring to a string.

Parameters

__pos	Index of first character of substring.
__n	Number of characters in substring.
__str	String to compare against.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n* characters starting at *__pos*. Returns an integer < 0 if the substring is ordered before *__str*, 0 if their values are equivalent, or > 0 if the substring is ordered after *__str*. Determines the effective length rlen of the strings to compare as the smallest of the length of the substring and *__str.size()*. The function then compares the two strings by calling traits::compare(substring.data(),str.data(),rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

compare() [3/12]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
int std::basic_string< _CharT, _Traits, _Allocator >::compare (
            size_type __pos,
            size_type __n,
            const basic_string< _CharT, _Traits, _Allocator > & __str) const  [inline]
```

Compare substring to a string.

Parameters

__pos	Index of first character of substring.
__n	Number of characters in substring.
__str	String to compare against.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n* characters starting at *__pos*. Returns an integer < 0 if the substring is ordered before *__str*, 0 if their values are equivalent, or > 0 if the substring is ordered after *__str*. Determines the effective length rlen of the strings to compare as the smallest of the length of the substring and *__str.size()*. The function then compares the two strings by calling traits::compare(substring.data(),str.data(),rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

compare() [4/12]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
int std::basic_string< _CharT, _Traits, _Allocator >::compare (
            size_type __pos,
            size_type __n1,
            const _CharT * __s) const  [inline], [nodiscard], [constexpr]
```

Compare substring to a C string.

Parameters

__pos	Index of first character of substring.
__n1	Number of characters in substring.
__s	C string to compare against.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n1* characters starting at *pos*. Returns an integer < 0 if the substring is ordered before *__s*, 0 if their values are equivalent, or > 0 if the substring is ordered after *__s*. Determines the effective length rlen of the strings to compare as the smallest of the length of the substring and the length of a string constructed from *__s*. The function then compares the two string by calling traits::compare(substring.data(),__s,rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

compare() [5/12]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
int std::basic_string< _CharT, _Traits, _Allocator >::compare (
            size_type __pos,
            size_type __n1,
            const _CharT * __s,
            size_type __n2) const  [inline], [nodiscard], [constexpr]
```

Compare substring against a character array.

Parameters

__pos	Index of first character of substring.
__n1	Number of characters in substring.
__s	character array to compare against.
__n2	Number of characters of s.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n1* characters starting at *__pos*. Form a string from the first *__n2* characters of *__s*. Returns an integer < 0 if this substring is ordered before the string from *__s*, 0 if their values are equivalent, or > 0 if this substring is ordered after the string from *__s*. Determines the effective length rlen of the strings to compare as the smallest of the length of the substring and *__n2*. The function then compares the two strings by calling traits↩ ::compare(substring.data(),s,rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.
NB: s must have at least n2 characters, '\0' has no special meaning.

compare() [6/12]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
int std::basic_string< _CharT, _Traits, _Allocator >::compare (
            size_type __pos,
            size_type __n1,
            const _CharT * __s,
            size_type __n2) const  [inline]
```

Compare substring against a character array.

Parameters

__pos	Index of first character of substring.
__n1	Number of characters in substring.
__s	character array to compare against.
__n2	Number of characters of s.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n1* characters starting at *__pos*. Form a string from the first *__n2* characters of *__s*. Returns an integer < 0 if this substring is ordered before the string from *__s*, 0 if their values are equivalent, or > 0 if this substring is ordered after the string from *__s*. Determines the effective length rlen of the strings to compare as the smallest of the length of the substring and *__n2*. The function then compares the two strings by calling traits↩ ::compare(substring.data(),s,rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.
NB: s must have at least n2 characters, '\0' has no special meaning.

compare() [7/12]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
int std::basic_string< _CharT, _Traits, _Allocator >::compare (
            size_type __pos1,
            size_type __n1,
            const basic_string< _CharT, _Traits, _Allocator > & __str,
            size_type __pos2,
            size_type __n2 = npos) const  [inline], [nodiscard], [constexpr]
```

Compare substring to a substring.

Parameters

__pos1	Index of first character of substring.
__n1	Number of characters in substring.
__str	String to compare against.
__pos2	Index of first character of substring of str.
__n2	Number of characters in substring of str.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n1* characters starting at *__pos1*. Form the substring of *__str* from the *__n2* characters starting at *__pos2*. Returns an integer < 0 if this substring is ordered before the substring of *__str*, 0 if their values are equivalent, or > 0 if this substring is ordered after the substring of *__str*. Determines the effective length rlen of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

compare() [8/12]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
int std::basic_string< _CharT, _Traits, _Allocator >::compare (
            size_type __pos1,
            size_type __n1,
            const basic_string< _CharT, _Traits, _Allocator > & __str,
            size_type __pos2,
            size_type __n2 = npos) const  [inline]
```

Compare substring to a substring.

Parameters

__pos1	Index of first character of substring.
__n1	Number of characters in substring.
__str	String to compare against.
__pos2	Index of first character of substring of str.
__n2	Number of characters in substring of str.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n1* characters starting at *__pos1*. Form the substring of *__str* from the *__n2* characters starting at *__pos2*. Returns an integer < 0 if this substring is ordered before the substring of *__str*, 0 if their values are equivalent, or > 0 if this substring is ordered after the substring of *__str*. Determines the effective length rlen of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

compare() [9/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
            size_type __pos,
            size_type __n,
            const basic_string< _CharT, _Traits, _Alloc > & __str) const  [inline], [nodiscard],
[constexpr], [inherited]
```

Compare substring to a string.

Parameters

__pos	Index of first character of substring.

Parameters

__n	Number of characters in substring.
__str	String to compare against.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n* characters starting at *__pos*. Returns an integer < 0 if the substring is ordered before *__str*, 0 if their values are equivalent, or > 0 if the substring is ordered after *__str*. Determines the effective length rlen of the strings to compare as the smallest of the length of the substring and *__str.size()*. The function then compares the two strings by calling traits::compare(substring.data(),str.data(),rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

compare() [10/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
            size_type __pos,
            size_type __n1,
            const _CharT * __s) const  [inline], [nodiscard], [constexpr], [inherited]
```

Compare substring to a C string.

Parameters

__pos	Index of first character of substring.
__n1	Number of characters in substring.
__s	C string to compare against.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n1* characters starting at *pos*. Returns an integer < 0 if the substring is ordered before *__s*, 0 if their values are equivalent, or > 0 if the substring is ordered after *__s*. Determines the effective length rlen of the strings to compare as the smallest of the length of the substring and the length of a string constructed from *__s*. The function then compares the two string by calling traits::compare(substring.data(),__s,rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

compare() [11/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
            size_type __pos,
            size_type __n1,
            const _CharT * __s,
            size_type __n2) const  [inline], [nodiscard], [constexpr], [inherited]
```

Compare substring against a character array.

Parameters

__pos	Index of first character of substring.
__n1	Number of characters in substring.

__s	character array to compare against.
__n2	Number of characters of s.

Returns

Integer $<$ 0, 0, or $>$ 0.

Form the substring of this string from the *__n1* characters starting at *__pos*. Form a string from the first *__n2* characters of *__s*. Returns an integer $<$ 0 if this substring is ordered before the string from *__s*, 0 if their values are equivalent, or $>$ 0 if this substring is ordered after the string from *__s*. Determines the effective length rlen of the strings to compare as the smallest of the length of the substring and *__n2*. The function then compares the two strings by calling traits::compare(substring.data(),s,rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.
NB: s must have at least n2 characters, '\0' has no special meaning.

compare() [12/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
            size_type __pos1,
            size_type __n1,
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos2,
            size_type __n2 = npos) const  [inline], [nodiscard], [constexpr], [inherited]
```

Compare substring to a substring.

Parameters

__pos1	Index of first character of substring.
__n1	Number of characters in substring.
__str	String to compare against.
__pos2	Index of first character of substring of str.
__n2	Number of characters in substring of str.

Returns

Integer $<$ 0, 0, or $>$ 0.

Form the substring of this string from the *__n1* characters starting at *__pos1*. Form the substring of *__str* from the *__n2* characters starting at *__pos2*. Returns an integer $<$ 0 if this substring is ordered before the substring of *__str*, 0 if their values are equivalent, or $>$ 0 if this substring is ordered after the substring of *__str*. Determines the effective length rlen of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

copy()

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::copy (
            _CharT * __s,
            size_type __n,
            size_type __pos = 0) const  [constexpr], [inherited]
```

Copy substring into C string.

Parameters

__s	C string to copy value into.
__n	Number of characters to copy.
__pos	Index of first character to copy.

Returns

Number of characters actually copied

Exceptions

std::out_of_range	If __pos > size().

Copies up to *__n* characters starting at *__pos* into the C string *__s*. If *__pos* is greater than size(), out_of_range is thrown.

data()

```
template<typename _CharT, typename _Traits, typename _Alloc>
_CharT * std::basic_string< _CharT, _Traits, _Alloc >::data ()  [inline], [nodiscard], [constexpr],
[noexcept], [inherited]
```

Return non-const pointer to contents.
This is a pointer to the character sequence held by the string. Modifying the characters in the sequence is allowed.

empty()

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
bool std::basic_string< _CharT, _Traits, _Allocator >::empty () const  [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns true if the string is empty. Equivalent to `*this == ""`.

erase() [1/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::erase (
            __const_iterator __first,
            __const_iterator __last)  [inline], [constexpr], [inherited]
```

Remove a range of characters.

Parameters

__first	Iterator referencing the first character to remove.
__last	Iterator referencing the end of the range.

Returns

Iterator referencing location of first after removal.

Removes the characters in the range [first,last) from this string. The value of the string doesn't change if an error is thrown.

erase() [2/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::erase (
            __const_iterator __position)  [inline], [constexpr], [inherited]
```

Remove one character.

Parameters

__position	Iterator referencing the character to remove.

Returns

iterator referencing same location after removal.

Removes the character at *__position* from this string. The value of the string doesn't change if an error is thrown.

erase() [3/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::iterator std::basic_string< _CharT, _Traits, _Alloc >↩
::erase (
            iterator __first,
            iterator __last)  [inherited]
```

Remove a range of characters.

Parameters

__first	Iterator referencing the first character to remove.
__last	Iterator referencing the end of the range.

Returns

Iterator referencing location of first after removal.

Removes the characters in the range [first,last) from this string. The value of the string doesn't change if an error is thrown.

erase() [4/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::erase (
            iterator __position)  [inline], [inherited]
```

Remove one character.

Parameters

__position	Iterator referencing the character to remove.

Returns

iterator referencing same location after removal.

Removes the character at *__position* from this string. The value of the string doesn't change if an error is thrown.

erase() [5/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::erase (
            size_type __pos = 0,
            size_type __n = npos)  [inline], [constexpr], [inherited]
```

Remove characters.

Parameters

__pos	Index of first character to remove (default 0).
__n	Number of characters to remove (default remainder).

Returns

Reference to this string.

Exceptions

std::out_of_range	If *pos* is beyond the end of this string.

Removes *__n* characters from this string starting at *__pos*. The length of the string is reduced by *__n*. If there are < *__n* characters to remove, the remainder of the string is truncated. If *__p* is beyond end of string, out_of_range is thrown. The value of the string doesn't change if an error is thrown.
Referenced by std::getline(), std::operator>>(), std::basic_string< _CharT >::pop_back(), and resize().

find() [1/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find (
            _CharT __c,
            size_type __pos = 0) const  [nodiscard], [constexpr], [noexcept]
```

Find position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for *__c* within this string. If found, returns the index where it was found. If not found, returns npos.

find() [2/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find (
            _CharT __c,
            size_type __pos = 0) const  [noexcept]
```

Find position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for *__c* within this string. If found, returns the index where it was found. If not found, returns npos.

find() [3/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [nodiscard], [constexpr], [noexcept]
```

Find position of a C substring.

Parameters

__s	C string to locate.
__pos	Index of character to search from.
__n	Number of characters from *s* to search for.

Returns

Index of start of first occurrence.

Starting from *__pos*, searches forward for the first *__n* characters in *__s* within this string. If found, returns the index where it begins. If not found, returns npos.

find() [4/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [noexcept]
```

Find position of a C substring.

Parameters

__s	C string to locate.
__pos	Index of character to search from.
__n	Number of characters from *s* to search for.

Returns

Index of start of first occurrence.

Starting from *__pos*, searches forward for the first *__n* characters in *__s* within this string. If found, returns the index where it begins. If not found, returns npos.

find() [5/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find (
            const _CharT * __s,
            size_type __pos = 0) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a C string.

Parameters

__s	C string to locate.
__pos	Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from *__pos*, searches forward for the value of *__s* within this string. If found, returns the index where it begins. If not found, returns npos.

find() [6/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find (
            const basic_string< _CharT, _Traits, _Allocator > & __str,
            size_type __pos = 0) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a string.

Parameters

__str	String to locate.
__pos	Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from *__pos*, searches forward for value of *__str* within this string. If found, returns the index where it begins. If not found, returns npos.

find() [7/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find (
            const basic_string< _CharT, _Traits, _Allocator > & __str,
            size_type __pos = 0) const  [inline], [noexcept]
```

Find position of a string.

Parameters

__str	String to locate.
__pos	Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from *__pos*, searches forward for value of *__str* within this string. If found, returns the index where it begins. If not found, returns npos.

find() [8/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find (
            _CharT __c,
            size_type __pos = 0) const  [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for *__c* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and std::size().

find() [9/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a C substring.

Parameters

__s	C string to locate.
__pos	Index of character to search from.
__n	Number of characters from *s* to search for.

Returns

Index of start of first occurrence.

Starting from *__pos*, searches forward for the first *__n* characters in *__s* within this string. If found, returns the index where it begins. If not found, returns npos.
References std::data(), npos, and std::size().
Referenced by std::basic_string< _CharT >::find(), std::basic_string< _CharT >::find(), std::basic_string< _CharT >::find(), and std::basic_string< _CharT >::find_first_of().

find() [10/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find (
            const _CharT * __s,
            size_type __pos = 0) const  [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a C string.

Parameters

__s	C string to locate.
__pos	Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from *__pos*, searches forward for the value of *__s* within this string. If found, returns the index where it begins. If not found, returns npos.

find() [11/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos = 0) const  [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a string.

Parameters

__str	String to locate.
__pos	Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from *__pos*, searches forward for value of *__str* within this string. If found, returns the index where it begins. If not found, returns npos.

find_first_not_of() [1/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_not_of (
            _CharT __c,
            size_type __pos = 0) const  [nodiscard], [constexpr], [noexcept]
```

Find position of a different character.

Parameters

__c	Character to avoid.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character other than *__c* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_not_of() [2/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_not_of (
            _CharT __c,
            size_type __pos = 0) const  [noexcept]
```

Find position of a different character.

Parameters

__c	Character to avoid.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character other than *__c* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_not_of() [3/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_not_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [nodiscard], [constexpr], [noexcept]
```

Find position of a character not in C substring.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search from.
__n	Number of characters from __s to consider.

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character not contained in the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_not_of() [4/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_not_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [noexcept]
```

Find position of a character not in C substring.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search from.
__n	Number of characters from __s to consider.

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character not contained in the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_not_of() [5/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_not_of (
            const _CharT * __s,
            size_type __pos = 0) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a character not in C string.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character not contained in *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_not_of() [6/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_not_of (
            const basic_string< _CharT, _Traits, _Allocator > & __str,
            size_type __pos = 0) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a character not in string.

Parameters

__str	String containing characters to avoid.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character not contained in *__str* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_not_of() [7/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_not_of (
            const basic_string< _CharT, _Traits, _Allocator > & __str,
            size_type __pos = 0) const  [inline], [noexcept]
```

Find position of a character not in string.

Parameters

__str	String containing characters to avoid.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character not contained in *__str* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_not_of() [8/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_first_not_of (
            _CharT __c,
            size_type __pos = 0) const  [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a different character.

Parameters

__c	Character to avoid.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character other than *__c* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and std::size().

find_first_not_of() [9/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_first_not_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a character not in C substring.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search from.
__n	Number of characters from __s to consider.

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character not contained in the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and std::size().

find_first_not_of() [10/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of (
            const _CharT * __s,
            size_type __pos = 0) const  [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a character not in C string.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character not contained in *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_not_of() [11/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos = 0) const  [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a character not in string.

Parameters

__str	String containing characters to avoid.

__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character not contained in *__str* within this string. If found, returns the index where it was found. If not found, returns npos.
Referenced by std::basic_string< _CharT >::find_first_not_of(), std::basic_string< _CharT >::find_first_not_of(), and std::basic_string< _CharT >::find_first_not_of().

find_first_of() [1/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_of (
            _CharT __c,
            size_type __pos = 0) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for the character *__c* within this string. If found, returns the index where it was found. If not found, returns npos.
Note: equivalent to find(__c, __pos).

find_first_of() [2/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_of (
            _CharT __c,
            size_type __pos = 0) const  [inline], [noexcept]
```

Find position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for the character *__c* within this string. If found, returns the index where it was found. If not found, returns npos.
Note: equivalent to find(__c, __pos).

find_first_of() [3/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [nodiscard], [constexpr], [noexcept]
```

Find position of a character of C substring.

Parameters

__s	String containing characters to locate.
__pos	Index of character to search from.
__n	Number of characters from s to search for.

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for one of the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_of() [4/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [noexcept]
```

Find position of a character of C substring.

Parameters

__s	String containing characters to locate.
__pos	Index of character to search from.
__n	Number of characters from s to search for.

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for one of the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_of() [5/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_of (
            const _CharT * __s,
            size_type __pos = 0) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a character of C string.

Parameters

__s	String containing characters to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for one of the characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_of() [6/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_of (
            const basic_string< _CharT, _Traits, _Allocator > & __str,
            size_type __pos = 0) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a character of string.

Parameters

__str	String containing characters to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for one of the characters of *__str* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_of() [7/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_of (
            const basic_string< _CharT, _Traits, _Allocator > & __str,
            size_type __pos = 0) const  [inline], [noexcept]
```

Find position of a character of string.

Parameters

__str	String containing characters to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for one of the characters of *__str* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_of() [8/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of (
            _CharT __c,
            size_type __pos = 0) const  [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for the character *__c* within this string. If found, returns the index where it was found. If not found, returns npos.
Note: equivalent to find(__c, __pos).

find_first_of() [9/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_first_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a character of C substring.

Parameters

__s	String containing characters to locate.
__pos	Index of character to search from.
__n	Number of characters from s to search for.

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for one of the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and std::size().

find_first_of() [10/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of (
            const _CharT * __s,
            size_type __pos = 0) const  [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a character of C string.

Parameters

__s	String containing characters to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for one of the characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_of() [11/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos = 0) const  [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a character of string.

Parameters

__str	String containing characters to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for one of the characters of *__str* within this string. If found, returns the index where it was found. If not found, returns npos.
Referenced by std::basic_string< _CharT >::find_first_of(), and std::basic_string< _CharT >::find_first_of().

find_last_not_of() [1/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_not_of (
            _CharT __c,
            size_type __pos = npos) const  [nodiscard], [constexpr], [noexcept]
```

Find last position of a different character.

Parameters

__c	Character to avoid.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character other than *__c* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_not_of() [2/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_not_of (
            _CharT __c,
            size_type __pos = npos) const  [noexcept]
```

Find last position of a different character.

Parameters

__c	Character to avoid.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character other than *__c* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_not_of() [3/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_not_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [nodiscard], [constexpr], [noexcept]
```

Find last position of a character not in C substring.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search back from.
__n	Number of characters from s to consider.

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character not contained in the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_not_of() [4/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_not_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [noexcept]
```

Find last position of a character not in C substring.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search back from.
__n	Number of characters from s to consider.

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character not contained in the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_not_of() [5/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_not_of (
            const _CharT * __s,
            size_type __pos = npos) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a character not in C string.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character not contained in *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_not_of() [6/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_not_of (
            const basic_string< _CharT, _Traits, _Allocator > & __str,
            size_type __pos = npos) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a character not in string.

Parameters

__str	String containing characters to avoid.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character not contained in *__str* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_not_of() [7/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_not_of (
            const basic_string< _CharT, _Traits, _Allocator > & __str,
            size_type __pos = npos) const  [inline], [noexcept]
```

Find last position of a character not in string.

Parameters

__str	String containing characters to avoid.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character not contained in *__str* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_not_of() [8/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_last_not_of (
            _CharT __c,
            size_type __pos = npos) const  [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a different character.

Parameters

__c	Character to avoid.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character other than *__c* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and std::size().

find_last_not_of() [9/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_last_not_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a character not in C substring.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search back from.
__n	Number of characters from s to consider.

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character not contained in the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and std::size().

find_last_not_of() [10/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (
            const _CharT * __s,
            size_type __pos = npos) const  [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a character not in C string.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character not contained in *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_not_of() [11/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos = npos) const  [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a character not in string.

Parameters

__str	String containing characters to avoid.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character not contained in *__str* within this string. If found, returns the index where it was found. If not found, returns npos.
Referenced by std::basic_string< _CharT >::find_last_not_of(), std::basic_string< _CharT >::find_last_not_of(), and std::basic_string< _CharT >::find_last_not_of().

find_last_of() [1/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_of (
            _CharT __c,
            size_type __pos = npos) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for *__c* within this string. If found, returns the index where it was found. If not found, returns npos.
Note: equivalent to rfind(__c, __pos).

find_last_of() [2/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_of (
            _CharT __c,
            size_type __pos = npos) const  [inline], [noexcept]
```

Find last position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for *__c* within this string. If found, returns the index where it was found. If not found, returns npos.
Note: equivalent to rfind(__c, __pos).

find_last_of() [3/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [nodiscard], [constexpr], [noexcept]
```

Find last position of a character of C substring.

Parameters

__s	C string containing characters to locate.
__pos	Index of character to search back from.
__n	Number of characters from s to search for.

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for one of the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_of() [4/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [noexcept]
```

Find last position of a character of C substring.

Parameters

__s	C string containing characters to locate.
__pos	Index of character to search back from.
__n	Number of characters from s to search for.

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for one of the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_of() [5/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_of (
            const _CharT * __s,
            size_type __pos = npos) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a character of C string.

Parameters

__s	C string containing characters to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for one of the characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_of() [6/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_of (
            const basic_string< _CharT, _Traits, _Allocator > & __str,
            size_type __pos = npos) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a character of string.

Parameters

__str	String containing characters to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for one of the characters of *__str* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_of() [7/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_of (
            const basic_string< _CharT, _Traits, _Allocator > & __str,
            size_type __pos = npos) const  [inline], [noexcept]
```

Find last position of a character of string.

Parameters

__str	String containing characters to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for one of the characters of *__str* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_of() [8/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of (
            _CharT __c,
            size_type __pos = npos) const  [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from __*pos*, searches backward for __*c* within this string. If found, returns the index where it was found. If not found, returns npos.
Note: equivalent to rfind(__c, __pos).

find_last_of() [9/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_last_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a character of C substring.

Parameters

__*s*	C string containing characters to locate.
__*pos*	Index of character to search back from.
__*n*	Number of characters from s to search for.

Returns

Index of last occurrence.

Starting from __*pos*, searches backward for one of the first __*n* characters of __*s* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and std::size().

find_last_of() [10/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of (
            const _CharT * __s,
            size_type __pos = npos) const  [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a character of C string.

Parameters

__*s*	C string containing characters to locate.
__*pos*	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from __*pos*, searches backward for one of the characters of __*s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_of() [11/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos = npos) const  [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a character of string.

Parameters

__str	String containing characters to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for one of the characters of *__str* within this string. If found, returns the index where it was found. If not found, returns npos.
Referenced by std::basic_string< _CharT >::find_last_of(), and std::basic_string< _CharT >::find_last_of().

front() [1/2]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
const_reference std::basic_string< _CharT, _Traits, _Allocator >::front () const  [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns a read-only (constant) reference to the data at the first element of the string.

front() [2/2]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
reference std::basic_string< _CharT, _Traits, _Allocator >::front ()  [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns a read/write reference to the data at the first element of the string.

get_allocator()

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
allocator_type std::basic_string< _CharT, _Traits, _Allocator >::get_allocator () const  [inline],
[nodiscard], [constexpr], [noexcept]
```

Return copy of allocator used to construct this string.

insert() [1/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::insert (
            __const_iterator __p,
            _CharT __c)  [inline], [constexpr], [inherited]
```

Insert one character.

Parameters

_↩ _p	Iterator referencing position in string to insert at.
_↩ _c	The character to insert.

Returns

Iterator referencing newly inserted char.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts character __c at position referenced by __p. If adding character causes the length to exceed max_size(), length_error is thrown. If __p is beyond end of string, out_of_range is thrown. The value of the string doesn't change if an error is thrown.

insert() [2/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
iterator std::basic_string< _CharT, _Traits, _Alloc >::insert (
            const_iterator __p,
            _InputIterator __beg,
            _InputIterator __end)  [inline], [constexpr], [inherited]
```

Insert a range of characters.

Parameters

__p	Const_iterator referencing location in string to insert at.
__beg	Start of range.
__end	End of range.

Returns

Iterator referencing the first inserted char.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts characters in range [beg,end). If adding characters causes the length to exceed max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

insert() [3/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::insert (
            const_iterator __p,
            initializer_list< _CharT > __l)  [inline], [constexpr], [inherited]
```

Insert an initializer_list of characters.

Parameters

_↩ _p	Iterator referencing location in string to insert at.
_↩ _l	The initializer_list of characters to insert.

Exceptions

std::length_error	If new length exceeds `max_size()`.

insert() [4/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::insert (
            const_iterator __p,
            size_type __n,
            _CharT __c)  [inline], [constexpr], [inherited]
```

Insert multiple characters.

Parameters

_↩ _p	Const_iterator referencing location in string to insert at.
_↩ _n	Number of characters to insert
_↩ _c	The character to insert.

Returns

Iterator referencing the first inserted char.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts *__n* copies of character *__c* starting at the position referenced by iterator *__p*. If adding characters causes the length to exceed max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.
Referenced by std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), and std::basic_string< _CharT >::insert().

insert() [5/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::insert (
            iterator __p,
            _CharT __c)  [inline], [inherited]
```

Insert one character.

Parameters

_↩ _p	Iterator referencing position in string to insert at.
_↩ _c	The character to insert.

Returns

Iterator referencing newly inserted char.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts character __c at position referenced by __p. If adding character causes the length to exceed max_size(), length_error is thrown. If __p is beyond end of string, out_of_range is thrown. The value of the string doesn't change if an error is thrown.

insert() [6/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator>
void std::basic_string< _CharT, _Traits, _Alloc >::insert (
             iterator __p,
             _InputIterator __beg,
             _InputIterator __end)  [inline], [inherited]
```

Insert a range of characters.

Parameters

__p	Iterator referencing location in string to insert at.
__beg	Start of range.
__end	End of range.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts characters in range [__beg,__end). If adding characters causes the length to exceed max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

insert() [7/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::insert (
             iterator __p,
             initializer_list< _CharT > __l)  [inline], [inherited]
```

Insert an initializer_list of characters.

Parameters

_↩ _p	Iterator referencing location in string to insert at.
_↩ _l	The initializer_list of characters to insert.

Exceptions

std::length_error	If new length exceeds `max_size()`.

insert() [8/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::insert (
            iterator __p,
            size_type __n,
            _CharT __c)  [inline], [inherited]
```

Insert multiple characters.

Parameters

_↩ _p	Iterator referencing location in string to insert at.
_↩ _n	Number of characters to insert
_↩ _c	The character to insert.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts *__n* copies of character *__c* starting at the position referenced by iterator *__p*. If adding characters causes the length to exceed max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

insert() [9/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::insert (
            size_type __pos,
            const _CharT * __s)  [inline], [constexpr], [inherited]
```

Insert a C string.

Parameters

__pos	Position in string to insert at.
__s	The C string to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.
std::out_of_range	If *pos* is beyond the end of this string.

Inserts the first *n* characters of *__s* starting at *__pos*. If adding characters causes the length to exceed max_size(), length_error is thrown. If *__pos* is beyond end(), out_of_range is thrown. The value of the string doesn't change if an error is thrown.

insert() [10/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::insert (
            size_type __pos,
            const _CharT * __s,
            size_type __n)  [inline], [constexpr], [inherited]
```

Insert a C substring.

Parameters

__pos	Position in string to insert at.
__s	The C string to insert.
__n	The number of characters to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.
std::out_of_range	If *__pos* is beyond the end of this string.

Inserts the first *__n* characters of *__s* starting at *__pos*. If adding characters causes the length to exceed max_size(), length_error is thrown. If *__pos* is beyond end(), out_of_range is thrown. The value of the string doesn't change if an error is thrown.

insert() [11/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::insert (
            size_type __pos,
            size_type __n,
            _CharT __c)  [inline], [constexpr], [inherited]
```

Insert multiple characters.

Parameters

__pos	Index in string to insert at.
__n	Number of characters to insert
__c	The character to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.
std::out_of_range	If *__pos* is beyond the end of this string.

Inserts *__n* copies of character *__c* starting at index *__pos*. If adding characters causes the length to exceed max_size(), length_error is thrown. If *__pos* > length(), out_of_range is thrown. The value of the string doesn't change if an error is thrown.

insert() [12/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::insert (
            size_type __pos1,
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline], [constexpr], [inherited]
```

Insert value of a string.

Parameters

__pos1	Position in string to insert at.
__str	The string to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts value of *__str* starting at *__pos1*. If adding characters causes the length to exceed max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

insert() [13/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::insert (
            size_type __pos1,
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos2,
            size_type __n = npos)  [inline], [constexpr], [inherited]
```

Insert a substring.

Parameters

__pos1	Position in string to insert at.
__str	The string to insert.
__pos2	Start of characters in str to insert.
__n	Number of characters to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.
std::out_of_range	If *pos1* > size() or *__pos2* > *str.size()*.

Starting at *pos1*, insert *__n* character of *__str* beginning with *__pos2*. If adding characters causes the length to exceed max_size(), length_error is thrown. If *__pos1* is beyond the end of this string or *__pos2* is beyond the end of *__str*, out_of_range is thrown. The value of the string doesn't change if an error is thrown.

length()

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::length () const  [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns the number of characters in the string, not including any null-termination.

max_size()

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::max_size () const  [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns the size() of the largest possible string.

operator+=()

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator+= (
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline], [constexpr], [inherited]
```

Append a string to this string.

Parameters

__str	The string to append.

Returns

Reference to this string.

operator[]() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string< _CharT, _Traits, _Alloc >::operator[] (
            size_type __pos)  [inline], [nodiscard], [constexpr], [inherited]
```

Subscript access to the data contained in the string.

Parameters

__pos	The index of the character to access.

Returns

Read/write reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out_of_range lookups are not defined. (For checked lookups see at().)

operator[]() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string< _CharT, _Traits, _Alloc >::operator[] (
            size_type __pos) const  [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Subscript access to the data contained in the string.

Parameters

__pos	The index of the character to access.

Returns

Read-only (constant) reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out_of_range lookups are not defined. (For checked lookups see at().)
Referenced by std::basic_string< _CharT >::back(), std::basic_string< _CharT >::back(), std::basic_string< _CharT >::back(), std::basic_string< _CharT >::front(), std::basic_string< _CharT >::front(), and std::basic_string< _CharT >::front().

replace() [1/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            __const_iterator __i1,
            __const_iterator __i2,
            const _CharT * __s)  [inline], [constexpr], [inherited]
```

Replace range of characters with C string.

Parameters

_↩ _i1	Iterator referencing start of range to replace.
_↩ _i2	Iterator referencing end of range to replace.
_↩ _s	C string value to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, the characters of __*s* are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [2/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            __const_iterator __i1,
            __const_iterator __i2,
            const _CharT * __s,
            size_type __n)  [inline], [constexpr], [inherited]
```

Replace range of characters with C substring.

Parameters

_↩ _*i1*	Iterator referencing start of range to replace.
_↩ _*i2*	Iterator referencing end of range to replace.
_↩ _*s*	C string value to insert.
_↩ _*n*	Number of characters from s to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, the first __*n* characters of __*s* are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [3/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            __const_iterator __i1,
            __const_iterator __i2,
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline], [constexpr], [inherited]
```

Replace range of characters with string.

Parameters

__i1	Iterator referencing start of range to replace.
__i2	Iterator referencing end of range to replace.
__str	String value to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, the value of *__str* is inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [4/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            __const_iterator __i1,
            __const_iterator __i2,
            size_type __n,
            _CharT __c)  [inline], [constexpr], [inherited]
```

Replace range of characters with multiple characters.

Parameters

_↩ _i1	Iterator referencing start of range to replace.
_↩ _i2	Iterator referencing end of range to replace.
_↩ _n	Number of characters to insert.
_↩ _c	Character to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, *__n* copies of *__c* are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [5/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            const_iterator __i1,
            const_iterator __i2,
            _InputIterator __k1,
            _InputIterator __k2)  [inline], [constexpr], [inherited]
```

Replace range of characters with range.

Parameters

__i1	Iterator referencing start of range to replace.
__i2	Iterator referencing end of range to replace.
__k1	Iterator referencing start of range to insert.
__k2	Iterator referencing end of range to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, characters in the range [__k1,__k2) are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [6/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            const_iterator __i1,
            const_iterator __i2,
            initializer_list< _CharT > __l)  [inline], [constexpr], [inherited]
```

Replace range of characters with initializer_list.

Parameters

_↩ _i1	Iterator referencing start of range to replace.
_↩ _i2	Iterator referencing end of range to replace.
_↩ _l	The initializer_list of characters to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, characters in the range [__k1,__k2) are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [7/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            iterator __i1,
            iterator __i2,
            _InputIterator __k1,
            _InputIterator __k2)  [inline], [inherited]
```

Replace range of characters with range.

Parameters

__i1	Iterator referencing start of range to replace.
__i2	Iterator referencing end of range to replace.
__k1	Iterator referencing start of range to insert.
__k2	Iterator referencing end of range to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, characters in the range [__k1,__k2) are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [8/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            iterator __i1,
            iterator __i2,
            const _CharT * __s)  [inline], [inherited]
```

Replace range of characters with C string.

Parameters

__i1	Iterator referencing start of range to replace.
__i2	Iterator referencing end of range to replace.
__s	C string value to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, the characters of __*s* are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [9/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            iterator __i1,
            iterator __i2,
            const _CharT * __s,
            size_type __n)  [inline], [inherited]
```

Replace range of characters with C substring.

Parameters

_↩ _*i1*	Iterator referencing start of range to replace.
_↩ _*i2*	Iterator referencing end of range to replace.
_↩ _*s*	C string value to insert.
_↩ _*n*	Number of characters from s to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, the first __*n* characters of __*s* are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [10/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            iterator __i1,
            iterator __i2,
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline], [inherited]
```

Replace range of characters with string.

Parameters

__i1	Iterator referencing start of range to replace.
__i2	Iterator referencing end of range to replace.
__str	String value to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, the value of *__str* is inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [11/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            iterator __i1,
            iterator __i2,
            initializer_list< _CharT > __l)  [inline], [inherited]
```

Replace range of characters with initializer_list.

Parameters

_↩ _i1	Iterator referencing start of range to replace.
_↩ _i2	Iterator referencing end of range to replace.
_↩ _l	The initializer_list of characters to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, characters in the range [__k1,__k2) are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [12/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            iterator __i1,
```

```
            iterator __i2,
            size_type __n,
            _CharT __c)  [inline], [inherited]
```

Replace range of characters with multiple characters.

Parameters

_↩ _i1	Iterator referencing start of range to replace.
_↩ _i2	Iterator referencing end of range to replace.
_↩ _n	Number of characters to insert.
_↩ _c	Character to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, __*n* copies of __*c* are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [13/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            size_type __pos,
            size_type __n,
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline], [constexpr], [inherited]
```

Replace characters with value from another string.

Parameters

__pos	Index of first character to replace.
__n	Number of characters to be replaced.
__str	String to insert.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *pos* is beyond the end of this string.
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__pos,__pos+__n) from this string. In place, the value of __*str* is inserted. If __*pos* is beyond end of string, out_of_range is thrown. If the length of the result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.
Referenced by std::basic_string< _CharT >::append(), std::basic_string< _CharT >::assign(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::replace(), std::basic_string< _CharT >::replace(), std::basic_string< _CharT >::replace(), std::basic_string< _CharT >::replace(), std::basic_string< _CharT >::replace(), std::basic_string< _CharT >::replace(), std::basic_string< _CharT >::replace(), std::basic_string< _CharT >::replace(), std::basic_string< _CharT >::replace(), and std::basic_string< _CharT >::replace().

replace() [14/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            size_type __pos,
            size_type __n1,
            const _CharT * __s)  [inline], [constexpr], [inherited]
```

Replace characters with value of a C string.

Parameters

__pos	Index of first character to replace.
__n1	Number of characters to be replaced.
__s	C string to insert.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *pos* > size().
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__pos,__pos + __n1) from this string. In place, the characters of __*s* are inserted. If __*pos* is beyond end of string, out_of_range is thrown. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [15/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::replace
(
            size_type __pos,
            size_type __n1,
            const _CharT * __s,
            size_type __n2)  [inline], [constexpr], [inherited]
```

Replace characters with value of a C substring.

Parameters

__pos	Index of first character to replace.
__n1	Number of characters to be replaced.
__s	C string to insert.
__n2	Number of characters from *s* to use.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *pos1* > size().
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__pos,__pos + __n1) from this string. In place, the first *__n2* characters of *__s* are inserted, or all of *__s* if *__n2* is too large. If *__pos* is beyond end of string, out_of_range is thrown. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [16/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            size_type __pos,
            size_type __n1,
            size_type __n2,
            _CharT __c)  [inline], [constexpr], [inherited]
```

Replace characters with multiple characters.

Parameters

__pos	Index of first character to replace.
__n1	Number of characters to be replaced.
__n2	Number of characters to insert.
__c	Character to insert.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *__pos* > size().
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [pos,pos + n1) from this string. In place, *__n2* copies of *__c* are inserted. If *__pos* is beyond end of string, out_of_range is thrown. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [17/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            size_type __pos1,
            size_type __n1,
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos2,
            size_type __n2 = npos)  [inline], [constexpr], [inherited]
```

Replace characters with value from another string.

Parameters

__pos1	Index of first character to replace.
__n1	Number of characters to be replaced.
__str	String to insert.
__pos2	Index of first character of str to use.
__n2	Number of characters from str to use.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *__pos1* > size() or *__pos2* > __str.size().
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__pos1,__pos1 + n) from this string. In place, the value of *__str* is inserted. If *__pos* is beyond end of string, out_of_range is thrown. If the length of the result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

reserve() [1/4]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
void std::basic_string< _CharT, _Traits, _Allocator >::reserve ()  [constexpr]
```

Equivalent to shrink_to_fit().

reserve() [2/4]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
void std::basic_string< _CharT, _Traits, _Allocator >::reserve (
            size_type __res_arg)  [constexpr]
```

Attempt to preallocate enough memory for specified number of characters.

Parameters

__res_arg	Number of characters required.

Exceptions

std::length_error	If *__res_arg* exceeds `max_size()`.

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than max_size(), length_error is thrown.
The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

reserve() [3/4]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
void std::basic_string< _CharT, _Traits, _Allocator >::reserve (
            size_type __res_arg)
```

Attempt to preallocate enough memory for specified number of characters.

Parameters

__res_arg	Number of characters required.

Exceptions

std::length_error	If *__res_arg* exceeds `max_size()`.

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than max_size(), length_error is thrown.
The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

reserve() [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::reserve (
            size_type __res_arg)  [constexpr], [inherited]
```

Attempt to preallocate enough memory for specified number of characters.

Parameters

__res_arg	Number of characters required.

Exceptions

std::length_error	If *__res_arg* exceeds `max_size()`.

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than max_size(), length_error is thrown.
The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

References capacity(), get_allocator(), and std::size().
Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >:: std::tr2::operator>>(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow().

resize() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::resize (
            size_type __n)  [inline], [constexpr], [inherited]
```

Resizes the string to the specified number of characters.

Parameters

_↩ _n	Number of characters the string should contain.

This function will resize the string to the specified length. If the new size is smaller than the string's current size the string is truncated, otherwise the string is extended and new characters are default-constructed. For basic types such as char, this means setting them to 0.

resize() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::resize (
            size_type __n,
            _CharT __c)  [constexpr], [inherited]
```

Resizes the string to the specified number of characters.

Parameters

_↩ _n	Number of characters the string should contain.
_↩ _c	Character to fill any new elements.

This function will resize the string to the specified number of characters. If the number is smaller than the string's current size the string is truncated, otherwise the string is extended and new elements are set to __c.
References append(), erase(), and size().
Referenced by std::basic_string< _CharT >::resize().

rfind() [1/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::rfind (
            _CharT __c,
            size_type __pos = npos) const  [nodiscard], [constexpr], [noexcept]
```

Find last position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for *__c* within this string. If found, returns the index where it was found. If not found, returns npos.

rfind() [2/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::rfind (
            _CharT __c,
            size_type __pos = npos) const  [noexcept]
```

Find last position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for *__c* within this string. If found, returns the index where it was found. If not found, returns npos.

rfind() [3/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::rfind (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [nodiscard], [constexpr], [noexcept]
```

Find last position of a C substring.

Parameters

__s	C string to locate.
__pos	Index of character to search back from.
__n	Number of characters from s to search for.

Returns

Index of start of last occurrence.

Starting from *__pos*, searches backward for the first *__n* characters in *__s* within this string. If found, returns the index where it begins. If not found, returns npos.

rfind() [4/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::rfind (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [noexcept]
```

Find last position of a C substring.

Parameters

__s	C string to locate.
__pos	Index of character to search back from.
__n	Number of characters from s to search for.

Returns

Index of start of last occurrence.

Starting from *__pos*, searches backward for the first *__n* characters in *__s* within this string. If found, returns the index where it begins. If not found, returns npos.

rfind() [5/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::rfind (
            const _CharT * __s,
            size_type __pos = npos) const  [inline], [nodiscard], [constexpr]
```

Find last position of a C string.

Parameters

__s	C string to locate.
__pos	Index of character to start search at (default end).

Returns

Index of start of last occurrence.

Starting from *__pos*, searches backward for the value of *__s* within this string. If found, returns the index where it begins. If not found, returns npos.

rfind() [6/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::rfind (
            const basic_string< _CharT, _Traits, _Allocator > & __str,
            size_type __pos = npos) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a string.

Parameters

__str	String to locate.
__pos	Index of character to search back from (default end).

Returns

Index of start of last occurrence.

Starting from *__pos*, searches backward for value of *__str* within this string. If found, returns the index where it begins. If not found, returns npos.

rfind() [7/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::rfind (
            const basic_string< _CharT, _Traits, _Allocator > & __str,
            size_type __pos = npos) const  [inline], [noexcept]
```

Find last position of a string.

Parameters

__str	String to locate.
__pos	Index of character to search back from (default end).

Returns

Index of start of last occurrence.

Starting from *__pos*, searches backward for value of *__str* within this string. If found, returns the index where it begins. If not found, returns npos.

rfind() [8/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::rfind (
            _CharT __c,
            size_type __pos = npos) const  [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for *__c* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and std::size().

rfind() [9/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::rfind (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a C substring.

Parameters

__s	C string to locate.
__pos	Index of character to search back from.
__n	Number of characters from s to search for.

Returns

Index of start of last occurrence.

Starting from *__pos*, searches backward for the first *__n* characters in *__s* within this string. If found, returns the index where it begins. If not found, returns npos.
References std::min(), npos, and std::size().

rfind() [10/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::rfind (
            const _CharT * __s,
            size_type __pos = npos) const  [inline], [nodiscard], [constexpr], [inherited]
```

Find last position of a C string.

Parameters

__s	C string to locate.
__pos	Index of character to start search at (default end).

Returns

Index of start of last occurrence.

Starting from *__pos*, searches backward for the value of *__s* within this string. If found, returns the index where it begins. If not found, returns npos.

rfind() [11/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::rfind (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos = npos) const  [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a string.

Parameters

__str	String to locate.

__pos	Index of character to search back from (default end).

Returns

Index of start of last occurrence.

Starting from *__pos*, searches backward for value of *__str* within this string. If found, returns the index where it begins. If not found, returns npos.
Referenced by std::basic_string< _CharT >::find_last_of(), std::basic_string< _CharT >::rfind(), std::basic_string< _CharT >::rfind(), std::basic_string< _CharT >::rfind(), and std::basic_string< _CharT >::rfind().

size()

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::size () const  [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns the number of characters in the string, not including any null-termination.

substr()

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string std::basic_string< _CharT, _Traits, _Alloc >::substr (
            size_type __pos = 0,
            size_type __n = npos) const  [inline], [nodiscard], [constexpr], [inherited]
```

Get a substring.

Parameters

__pos	Index of first character (default 0).
__n	Number of characters in substring (default remainder).

Returns

The new string.

Exceptions

std::out_of_range	If __pos > size().

Construct and return a new string using the *__n* characters starting at *__pos*. If the string is too short, use the remainder of the characters. If *__pos* is beyond the end of the string, out_of_range is thrown.

swap()

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::swap (
            basic_string< _CharT, _Traits, _Alloc > & __s)  [constexpr], [noexcept], [inherited]
```

Swap contents with another string.

Parameters

_↩ _s	String to swap with.

Exchanges the contents of this string with that of __s in constant time.
References basic_string(), and get_allocator().
Referenced by std::basic_string< _CharT >::assign(), and std::basic_string< _CharT >::operator=().

5.232.3 Member Data Documentation

npos

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
const size_type std::basic_string< _CharT, _Traits, _Allocator >::npos  [static]
```

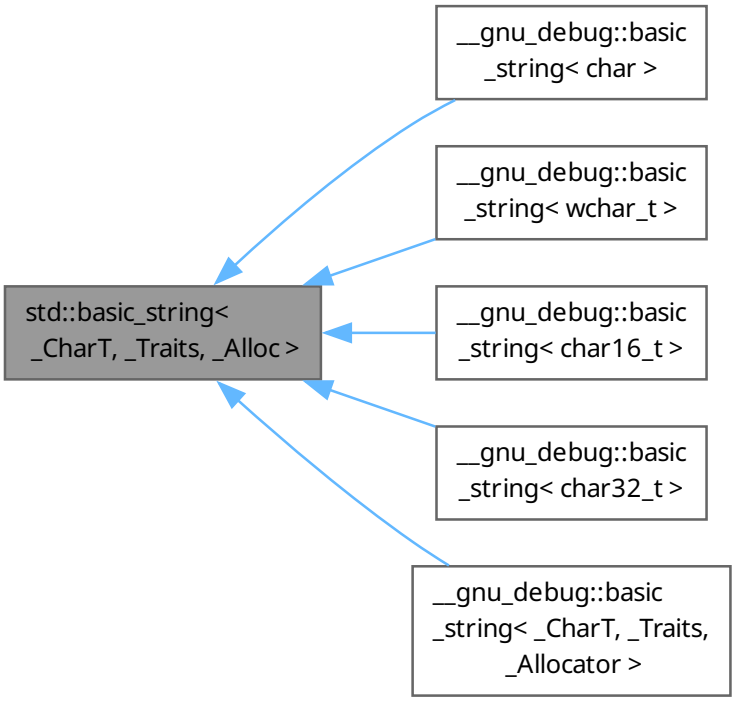
Value returned by various member functions when they fail.
The documentation for this class was generated from the following file:

- debug/string

5.233 std::basic_string< _CharT, _Traits, _Alloc > Class Template Reference

```
#include <string>
```

Inheritance diagram for std::basic_string< _CharT, _Traits, _Alloc >:



Public Types

- typedef _Char_alloc_type **allocator_type**
- typedef _Alloc **allocator_type**
- typedef __gnu_cxx::__normal_iterator< const_pointer, basic_string > **const_iterator**
- typedef __gnu_cxx::__normal_iterator< const_pointer, basic_string > **const_iterator**
- typedef _Alloc_traits::const_pointer **const_pointer**
- typedef _CharT_alloc_traits::const_pointer **const_pointer**
- typedef _Alloc_traits::const_reference **const_reference**
- typedef const value_type & **const_reference**
- typedef std::reverse_iterator< const_iterator > **const_reverse_iterator**
- typedef std::reverse_iterator< const_iterator > **const_reverse_iterator**
- typedef _Alloc_traits::difference_type **difference_type**
- typedef _CharT_alloc_traits::difference_type **difference_type**
- typedef __gnu_cxx::__normal_iterator< pointer, basic_string > **iterator**
- typedef __gnu_cxx::__normal_iterator< pointer, basic_string > **iterator**
- typedef _Alloc_traits::pointer **pointer**
- typedef _CharT_alloc_traits::pointer **pointer**
- typedef _Alloc_traits::reference **reference**
- typedef value_type & **reference**
- typedef std::reverse_iterator< iterator > **reverse_iterator**
- typedef std::reverse_iterator< iterator > **reverse_iterator**
- typedef _Alloc_traits::size_type **size_type**
- typedef _CharT_alloc_traits::size_type **size_type**
- typedef _Traits **traits_type**
- typedef _Traits **traits_type**
- typedef _Traits::char_type **value_type**
- typedef _Traits::char_type **value_type**

Public Member Functions

- basic_string () noexcept
- constexpr basic_string () noexcept(/*conditional */)
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  constexpr basic_string (_InputIterator __beg, _InputIterator __end, const _Alloc &__a=_Alloc())
- template<class _InputIterator>
  basic_string (_InputIterator __beg, _InputIterator __end, const _Alloc &__a=_Alloc())
- constexpr basic_string (basic_string &&__str) noexcept
- basic_string (basic_string &&__str) noexcept
- **basic_string** (basic_string &&__str, const _Alloc &__a)
- constexpr **basic_string** (basic_string &&__str, const _Alloc &__a) noexcept(_Alloc_traits::_S_always_equal())
- basic_string (const _Alloc &__a)
- constexpr basic_string (const _Alloc &__a) noexcept
- template<typename = _RequireAllocator<_Alloc>>
  constexpr basic_string (const _CharT *__s, const _Alloc &__a=_Alloc())
- template<typename = _RequireAllocator<_Alloc>>
  basic_string (const _CharT *__s, const _Alloc &__a=_Alloc())
- constexpr basic_string (const _CharT *__s, size_type __n, const _Alloc &__a=_Alloc())
- basic_string (const _CharT *__s, size_type __n, const _Alloc &__a=_Alloc())
- constexpr basic_string (const basic_string &__str)
- basic_string (const basic_string &__str)
- constexpr **basic_string** (const basic_string &__str, const _Alloc &__a)

- **basic_string** (const basic_string &__str, const _Alloc &__a)
- constexpr basic_string (const basic_string &__str, size_type __pos, const _Alloc &__a=_Alloc())
- basic_string (const basic_string &__str, size_type __pos, const _Alloc &__a=_Alloc())
- constexpr basic_string (const basic_string &__str, size_type __pos, size_type __n)
- basic_string (const basic_string &__str, size_type __pos, size_type __n)
- constexpr basic_string (const basic_string &__str, size_type __pos, size_type __n, const _Alloc &__a)
- basic_string (const basic_string &__str, size_type __pos, size_type __n, const _Alloc &__a)
- constexpr basic_string (initializer_list< _CharT > __l, const _Alloc &__a=_Alloc())
- basic_string (initializer_list< _CharT > __l, const _Alloc &__a=_Alloc())
- template<typename = _RequireAllocator<_Alloc>>
  constexpr basic_string (size_type __n, _CharT __c, const _Alloc &__a=_Alloc())
- basic_string (size_type __n, _CharT __c, const _Alloc &__a=_Alloc())
- constexpr ~basic_string ()
- ~basic_string () noexcept
- template<typename _Operation>
  constexpr void __resize_and_overwrite (size_type __n, _Operation __op)
- template<typename _Operation>
  void __resize_and_overwrite (size_type __n, _Operation __op)
- template<typename _InputIterator>
  basic_string< _CharT, _Traits, _Alloc > & **_M_replace_dispatch** (iterator __i1, iterator __i2, _InputIterator __k1, _InputIterator __k2, __false_type)
- template<typename _InIterator>
  _CharT * **_S_construct** (_InIterator __beg, _InIterator __end, const _Alloc &__a, forward_iterator_tag)
- template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  constexpr basic_string & append (_InputIterator __first, _InputIterator __last)
- template<class _InputIterator>
  basic_string & append (_InputIterator __first, _InputIterator __last)
- constexpr basic_string & append (const _CharT *__s)
- basic_string & append (const _CharT *__s)
- constexpr basic_string & append (const _CharT *__s, size_type __n)
- basic_string & append (const _CharT *__s, size_type __n)
- constexpr basic_string & append (const basic_string &__str)
- basic_string & append (const basic_string &__str)
- constexpr basic_string & append (const basic_string &__str, size_type __pos, size_type __n=npos)
- basic_string & append (const basic_string &__str, size_type __pos, size_type __n=npos)
- constexpr basic_string & append (initializer_list< _CharT > __l)
- basic_string & append (initializer_list< _CharT > __l)
- constexpr basic_string & append (size_type __n, _CharT __c)
- basic_string & append (size_type __n, _CharT __c)
- template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  constexpr basic_string & assign (_InputIterator __first, _InputIterator __last)
- template<class _InputIterator>
  basic_string & assign (_InputIterator __first, _InputIterator __last)
- constexpr basic_string & assign (basic_string &&__str) noexcept(_Alloc_traits::_S_nothrow_move())
- basic_string & assign (basic_string &&__str) noexcept(allocator_traits< _Alloc >::is_always_equal::value)
- constexpr basic_string & assign (const _CharT *__s)
- basic_string & assign (const _CharT *__s)
- constexpr basic_string & assign (const _CharT *__s, size_type __n)
- basic_string & assign (const _CharT *__s, size_type __n)
- constexpr basic_string & assign (const basic_string &__str)
- basic_string & assign (const basic_string &__str)

- constexpr basic_string & assign (const basic_string &__str, size_type __pos, size_type __n=npos)
- basic_string & assign (const basic_string &__str, size_type __pos, size_type __n=npos)
- constexpr basic_string & assign (initializer_list< _CharT > __l)
- basic_string & assign (initializer_list< _CharT > __l)
- constexpr basic_string & assign (size_type __n, _CharT __c)
- basic_string & assign (size_type __n, _CharT __c)
- constexpr reference at (size_type __n)
- reference at (size_type __n)
- constexpr const_reference at (size_type __n) const
- const_reference at (size_type __n) const
- reference back ()
- constexpr const_reference back () const noexcept
- const_reference back () const noexcept
- constexpr reference back () noexcept
- iterator begin ()
- constexpr const_iterator begin () const noexcept
- const_iterator begin () const noexcept
- constexpr iterator begin () noexcept
- constexpr const _CharT * c_str () const noexcept
- const _CharT * c_str () const noexcept
- constexpr size_type capacity () const noexcept
- size_type capacity () const noexcept
- constexpr const_iterator cbegin () const noexcept
- const_iterator cbegin () const noexcept
- constexpr const_iterator cend () const noexcept
- const_iterator cend () const noexcept
- constexpr void clear () noexcept
- void clear () noexcept
- constexpr int compare (const _CharT *__s) const noexcept
- int compare (const _CharT *__s) const noexcept
- constexpr int compare (const basic_string &__str) const
- int compare (const basic_string &__str) const
- constexpr int compare (size_type __pos, size_type __n, const basic_string &__str) const
- int compare (size_type __pos, size_type __n, const basic_string &__str) const
- constexpr int compare (size_type __pos, size_type __n1, const _CharT *__s) const
- int compare (size_type __pos, size_type __n1, const _CharT *__s) const
- constexpr int compare (size_type __pos, size_type __n1, const _CharT *__s, size_type __n2) const
- int compare (size_type __pos, size_type __n1, const _CharT *__s, size_type __n2) const
- constexpr int compare (size_type __pos1, size_type __n1, const basic_string &__str, size_type __pos2, size_type __n2=npos) const
- int compare (size_type __pos1, size_type __n1, const basic_string &__str, size_type __pos2, size_type __n2=npos) const
- constexpr size_type copy (_CharT *__s, size_type __n, size_type __pos=0) const
- size_type copy (_CharT *__s, size_type __n, size_type __pos=0) const
- constexpr const_reverse_iterator crbegin () const noexcept
- const_reverse_iterator crbegin () const noexcept
- constexpr const_reverse_iterator crend () const noexcept
- const_reverse_iterator crend () const noexcept
- constexpr const _CharT * data () const noexcept
- const _CharT * data () const noexcept
- constexpr _CharT * data () noexcept

- _CharT * data () noexcept(false)
- constexpr bool empty () const noexcept
- bool empty () const noexcept
- iterator end ()
- constexpr const_iterator end () const noexcept
- const_iterator end () const noexcept
- constexpr iterator end () noexcept
- constexpr bool **ends_with** (_CharT __x) const noexcept
- bool **ends_with** (_CharT __x) const noexcept
- constexpr bool **ends_with** (basic_string_view< _CharT, _Traits > __x) const noexcept
- bool **ends_with** (basic_string_view< _CharT, _Traits > __x) const noexcept
- constexpr bool **ends_with** (const _CharT *__x) const noexcept
- bool **ends_with** (const _CharT *__x) const noexcept
- constexpr iterator erase (__const_iterator __first, __const_iterator __last)
- constexpr iterator erase (__const_iterator __position)
- iterator erase (iterator __first, iterator __last)
- iterator erase (iterator __position)
- constexpr basic_string & erase (size_type __pos=0, size_type __n=npos)
- basic_string & erase (size_type __pos=0, size_type __n=npos)
- constexpr size_type find (_CharT __c, size_type __pos=0) const noexcept
- size_type find (_CharT __c, size_type __pos=0) const noexcept
- constexpr size_type find (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- size_type find (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type find (const _CharT *__s, size_type __pos=0) const noexcept
- size_type find (const _CharT *__s, size_type __pos=0) const noexcept
- constexpr size_type find (const basic_string &__str, size_type __pos=0) const noexcept
- size_type find (const basic_string &__str, size_type __pos=0) const noexcept
- constexpr size_type find_first_not_of (_CharT __c, size_type __pos=0) const noexcept
- size_type find_first_not_of (_CharT __c, size_type __pos=0) const noexcept
- constexpr size_type find_first_not_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- size_type find_first_not_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type find_first_not_of (const _CharT *__s, size_type __pos=0) const noexcept
- size_type find_first_not_of (const _CharT *__s, size_type __pos=0) const noexcept
- constexpr size_type find_first_not_of (const basic_string &__str, size_type __pos=0) const noexcept
- size_type find_first_not_of (const basic_string &__str, size_type __pos=0) const noexcept
- constexpr size_type find_first_of (_CharT __c, size_type __pos=0) const noexcept
- size_type find_first_of (_CharT __c, size_type __pos=0) const noexcept
- constexpr size_type find_first_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- size_type find_first_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type find_first_of (const _CharT *__s, size_type __pos=0) const noexcept
- size_type find_first_of (const _CharT *__s, size_type __pos=0) const noexcept
- constexpr size_type find_first_of (const basic_string &__str, size_type __pos=0) const noexcept
- size_type find_first_of (const basic_string &__str, size_type __pos=0) const noexcept
- constexpr size_type find_last_not_of (_CharT __c, size_type __pos=npos) const noexcept
- size_type find_last_not_of (_CharT __c, size_type __pos=npos) const noexcept
- constexpr size_type find_last_not_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- size_type find_last_not_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type find_last_not_of (const _CharT *__s, size_type __pos=npos) const noexcept
- size_type find_last_not_of (const _CharT *__s, size_type __pos=npos) const noexcept
- constexpr size_type find_last_not_of (const basic_string &__str, size_type __pos=npos) const noexcept
- size_type find_last_not_of (const basic_string &__str, size_type __pos=npos) const noexcept

- constexpr size_type find_last_of (_CharT __c, size_type __pos=npos) const noexcept
- size_type find_last_of (_CharT __c, size_type __pos=npos) const noexcept
- constexpr size_type find_last_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- size_type find_last_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type find_last_of (const _CharT *__s, size_type __pos=npos) const noexcept
- size_type find_last_of (const _CharT *__s, size_type __pos=npos) const noexcept
- constexpr size_type find_last_of (const basic_string &__str, size_type __pos=npos) const noexcept
- size_type find_last_of (const basic_string &__str, size_type __pos=npos) const noexcept
- reference front ()
- constexpr const_reference front () const noexcept
- const_reference front () const noexcept
- constexpr reference front () noexcept
- constexpr allocator_type get_allocator () const noexcept
- allocator_type get_allocator () const noexcept
- constexpr iterator insert (__const_iterator __p, _CharT __c)
- template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  constexpr iterator insert (const_iterator __p, _InputIterator __beg, _InputIterator __end)
- constexpr iterator insert (const_iterator __p, initializer_list< _CharT > __l)
- constexpr iterator insert (const_iterator __p, size_type __n, _CharT __c)
- iterator insert (iterator __p, _CharT __c)
- template<class _InputIterator>
  void insert (iterator __p, _InputIterator __beg, _InputIterator __end)
- void insert (iterator __p, initializer_list< _CharT > __l)
- void insert (iterator __p, size_type __n, _CharT __c)
- constexpr basic_string & insert (size_type __pos, const _CharT *__s)
- basic_string & insert (size_type __pos, const _CharT *__s)
- constexpr basic_string & insert (size_type __pos, const _CharT *__s, size_type __n)
- basic_string & insert (size_type __pos, const _CharT *__s, size_type __n)
- constexpr basic_string & insert (size_type __pos, size_type __n, _CharT __c)
- basic_string & insert (size_type __pos, size_type __n, _CharT __c)
- constexpr basic_string & insert (size_type __pos1, const basic_string &__str)
- basic_string & insert (size_type __pos1, const basic_string &__str)
- constexpr basic_string & insert (size_type __pos1, const basic_string &__str, size_type __pos2, size_type __n=npos)
- basic_string & insert (size_type __pos1, const basic_string &__str, size_type __pos2, size_type __n=npos)
- constexpr size_type length () const noexcept
- size_type length () const noexcept
- constexpr size_type max_size () const noexcept
- size_type max_size () const noexcept
- constexpr basic_string & operator+= (_CharT __c)
- basic_string & operator+= (_CharT __c)
- constexpr basic_string & operator+= (const _CharT *__s)
- basic_string & operator+= (const _CharT *__s)
- constexpr basic_string & operator+= (const basic_string &__str)
- basic_string & operator+= (const basic_string &__str)
- constexpr basic_string & operator+= (initializer_list< _CharT > __l)
- basic_string & operator+= (initializer_list< _CharT > __l)
- constexpr basic_string & operator= (_CharT __c)
- basic_string & operator= (_CharT __c)
- basic_string & operator= (basic_string &&__str) noexcept(/*conditional */)
- constexpr basic_string & operator= (basic_string &&__str) noexcept(_Alloc_traits::_S_nothrow_move())

- constexpr basic_string & operator= (const _CharT *__s)
- basic_string & operator= (const _CharT *__s)
- constexpr basic_string & operator= (const basic_string &__str)
- basic_string & operator= (const basic_string &__str)
- constexpr basic_string & operator= (initializer_list< _CharT > __l)
- basic_string & operator= (initializer_list< _CharT > __l)
- constexpr reference operator[] (size_type __pos)
- reference operator[] (size_type __pos)
- constexpr const_reference operator[] (size_type __pos) const noexcept
- const_reference operator[] (size_type __pos) const noexcept
- void pop_back ()
- constexpr void pop_back () noexcept
- constexpr void push_back (_CharT __c)
- void push_back (_CharT __c)
- reverse_iterator rbegin ()
- constexpr const_reverse_iterator rbegin () const noexcept
- const_reverse_iterator rbegin () const noexcept
- constexpr reverse_iterator rbegin () noexcept
- reverse_iterator rend ()
- constexpr const_reverse_iterator rend () const noexcept
- const_reverse_iterator rend () const noexcept
- constexpr reverse_iterator rend () noexcept
- constexpr basic_string & **replace** (__const_iterator __i1, __const_iterator __i2, _CharT *__k1, _CharT *__k2)
- constexpr basic_string & **replace** (__const_iterator __i1, __const_iterator __i2, const _CharT *__k1, const _CharT *__k2)
- constexpr basic_string & replace (__const_iterator __i1, __const_iterator __i2, const _CharT *__s)
- constexpr basic_string & replace (__const_iterator __i1, __const_iterator __i2, const _CharT *__s, size_type __n)
- constexpr basic_string & replace (__const_iterator __i1, __const_iterator __i2, const basic_string &__str)
- constexpr basic_string & **replace** (__const_iterator __i1, __const_iterator __i2, const_iterator __k1, const_iterator __k2)
- constexpr basic_string & **replace** (__const_iterator __i1, __const_iterator __i2, iterator __k1, iterator __k2)
- constexpr basic_string & replace (__const_iterator __i1, __const_iterator __i2, size_type __n, _CharT __c)
- template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  constexpr basic_string & replace (const_iterator __i1, const_iterator __i2, _InputIterator __k1, _InputIterator __k2)
- constexpr basic_string & replace (const_iterator __i1, const_iterator __i2, initializer_list< _CharT > __l)
- basic_string & **replace** (iterator __i1, iterator __i2, _CharT *__k1, _CharT *__k2)
- template<class _InputIterator>
  basic_string & replace (iterator __i1, iterator __i2, _InputIterator __k1, _InputIterator __k2)
- basic_string & **replace** (iterator __i1, iterator __i2, const _CharT *__k1, const _CharT *__k2)
- basic_string & replace (iterator __i1, iterator __i2, const _CharT *__s)
- basic_string & replace (iterator __i1, iterator __i2, const _CharT *__s, size_type __n)
- basic_string & replace (iterator __i1, iterator __i2, const basic_string &__str)
- basic_string & **replace** (iterator __i1, iterator __i2, const_iterator __k1, const_iterator __k2)
- basic_string & replace (iterator __i1, iterator __i2, initializer_list< _CharT > __l)
- basic_string & **replace** (iterator __i1, iterator __i2, iterator __k1, iterator __k2)
- basic_string & replace (iterator __i1, iterator __i2, size_type __n, _CharT __c)
- constexpr basic_string & replace (size_type __pos, size_type __n, const basic_string &__str)
- basic_string & replace (size_type __pos, size_type __n, const basic_string &__str)
- constexpr basic_string & replace (size_type __pos, size_type __n1, const _CharT *__s)
- basic_string & replace (size_type __pos, size_type __n1, const _CharT *__s)
- constexpr basic_string & replace (size_type __pos, size_type __n1, const _CharT *__s, size_type __n2)

- basic_string & replace (size_type __pos, size_type __n1, const _CharT *__s, size_type __n2)
- constexpr basic_string & replace (size_type __pos, size_type __n1, size_type __n2, _CharT __c)
- basic_string & replace (size_type __pos, size_type __n1, size_type __n2, _CharT __c)
- constexpr basic_string & replace (size_type __pos1, size_type __n1, const basic_string &__str, size_type __pos2, size_type __n2=npos)
- basic_string & replace (size_type __pos1, size_type __n1, const basic_string &__str, size_type __pos2, size_type __n2=npos)
- constexpr void reserve ()
- void reserve ()
- constexpr void reserve (size_type __res_arg)
- void reserve (size_type __res_arg)
- constexpr void resize (size_type __n)
- void resize (size_type __n)
- constexpr void resize (size_type __n, _CharT __c)
- void resize (size_type __n, _CharT __c)
- constexpr size_type rfind (_CharT __c, size_type __pos=npos) const noexcept
- size_type rfind (_CharT __c, size_type __pos=npos) const noexcept
- constexpr size_type rfind (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- size_type rfind (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type rfind (const _CharT *__s, size_type __pos=npos) const
- size_type rfind (const _CharT *__s, size_type __pos=npos) const noexcept
- constexpr size_type rfind (const basic_string &__str, size_type __pos=npos) const noexcept
- size_type rfind (const basic_string &__str, size_type __pos=npos) const noexcept
- constexpr void shrink_to_fit () noexcept
- void shrink_to_fit () noexcept
- constexpr size_type size () const noexcept
- size_type size () const noexcept
- constexpr bool **starts_with** (_CharT __x) const noexcept
- bool **starts_with** (_CharT __x) const noexcept
- constexpr bool **starts_with** (basic_string_view< _CharT, _Traits > __x) const noexcept
- bool **starts_with** (basic_string_view< _CharT, _Traits > __x) const noexcept
- constexpr bool **starts_with** (const _CharT *__x) const noexcept
- bool **starts_with** (const _CharT *__x) const noexcept
- constexpr basic_string substr (size_type __pos=0, size_type __n=npos) const
- basic_string substr (size_type __pos=0, size_type __n=npos) const
- constexpr void swap (basic_string &__s) noexcept
- void swap (basic_string &__s) noexcept(/*conditional */)

Static Public Attributes

- static const size_type npos

Protected Types

- typedef const_iterator **__const_iterator**
- typedef iterator **__const_iterator**

Friends

- template<typename, typename, typename>
  class **basic_stringbuf**

5.233.1 Detailed Description

template<typename _CharT, typename _Traits, typename _Alloc>
class std::basic_string< _CharT, _Traits, _Alloc >

Managing sequences of characters and character-like objects.

Since

C++98

Template Parameters

_CharT	Type of character
_Traits	Traits for character type, defaults to char_traits<_CharT>.
_Alloc	Allocator type, defaults to allocator<_CharT>.

Meets the requirements of a container, a reversible container, and a sequence. Of the optional sequence requirements, only push_back, at, and array access are supported.

Since

C++98

Template Parameters

_CharT	Type of character
_Traits	Traits for character type, defaults to char_traits<_CharT>.
_Alloc	Allocator type, defaults to allocator<_CharT>.

Meets the requirements of a container, a reversible container, and a sequence. Of the optional sequence requirements, only push_back, at, and array access are supported.

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

Documentation? What's that? Nathan Myers ncm@cantrip.org.
A string looks like this:

```
                                    [_Rep]
                                    _M_length
[basic_string<char_type>]           _M_capacity
_M_dataplus                         _M_refcount
_M_p ---------------->              unnamed array of char_type
```

Where the _M_p points to the first character in the string, and you cast it to a pointer-to-_Rep and subtract 1 to get a pointer to the header.
This approach has the enormous advantage that a string object requires only one allocation. All the ugliness is confined within a single pair of inline functions, which each compile to a single *add* instruction: _Rep::_M_data(), and string::_M_rep(); and the allocation function which gets a block of raw bytes and with room enough and constructs a _Rep object at the front.
The reason you want _M_data pointing to the character array and not the _Rep is so that the debugger can see the string contents. (Probably we should add a non-inline member to get the _Rep for the debugger to use, so users can check the actual string length.)
Note that the _Rep object is a POD so that you can have a static *empty string* _Rep object already *constructed* before static constructors have run. The reference-count encoding is chosen so that a 0 indicates one reference, so you never try to destroy the empty-string _Rep object.
All but the last paragraph is considered pretty conventional for a Copy-On-Write C++ string implementation.

5.233.2 Constructor & Destructor Documentation

basic_string() [1/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string ()  [inline], [constexpr], [noexcept]
```

Default constructor creates an empty string.
Referenced by std::basic_string< _CharT >::assign(), std::basic_string< _CharT >::assign(), std::basic_string< _CharT >::substr(), and swap().

basic_string() [2/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            const _Alloc & __a)  [inline], [explicit], [constexpr], [noexcept]
```

Construct an empty string using allocator *a*.

basic_string() [3/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline], [constexpr]
```

Construct string with copy of value of *__str*.

Parameters

__str	Source string.

basic_string() [4/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos,
            const _Alloc & __a = _Alloc())  [inline], [constexpr]
```

Construct string as copy of a substring.

Parameters

__str	Source string.
__pos	Index of first character to copy from.
__a	Allocator to use.

basic_string() [5/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos,
            size_type __n)  [inline], [constexpr]
```

Construct string as copy of a substring.

Parameters

__str	Source string.
__pos	Index of first character to copy from.
__n	Number of characters to copy.

basic_string() [6/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos,
            size_type __n,
            const _Alloc & __a)  [inline], [constexpr]
```

Construct string as copy of a substring.

Parameters

__str	Source string.
__pos	Index of first character to copy from.
__n	Number of characters to copy.
__a	Allocator to use.

basic_string() [7/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            const _CharT * __s,
            size_type __n,
            const _Alloc & __a = _Alloc())  [inline], [constexpr]
```

Construct string initialized by a character array.

Parameters

_↩ _s	Source character array.
_↩ _n	Number of characters to copy.
_↩ _a	Allocator to use (default is default allocator).

NB: *__s* must have at least *__n* characters, '\0' has no special meaning.

basic_string() [8/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<typename = _RequireAllocator<_Alloc>>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            const _CharT * __s,
            const _Alloc & __a = _Alloc())  [inline], [constexpr]
```

Construct string as copy of a C string.

Parameters

_↩ _s	Source C string.
_↩ _a	Allocator to use (default is default allocator).

basic_string() [9/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<typename = _RequireAllocator<_Alloc>>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            size_type __n,
            _CharT __c,
            const _Alloc & __a = _Alloc())  [inline], [constexpr]
```

Construct string as multiple characters.

Parameters

_↩ _n	Number of characters.
_↩ _c	Character to use.
_↩ _a	Allocator to use (default is default allocator).

basic_string() [10/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            basic_string< _CharT, _Traits, _Alloc > && __str)  [inline], [constexpr], [noexcept]
```

Move construct string.

Parameters

__str	Source string.

The newly-created string contains the exact contents of __str. __str is a valid, but unspecified string.

basic_string() [11/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            initializer_list< _CharT > __l,
            const _Alloc & __a = _Alloc())  [inline], [constexpr]
```

Construct string from an initializer list.

Parameters

_↩ _l	std::initializer_list of characters.

_↩ _a	Allocator to use (default is default allocator).

basic_string() [12/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            _InputIterator __beg,
            _InputIterator __end,
            const _Alloc & __a = _Alloc())  [inline], [constexpr]
```

Construct string as copy of a range.

Parameters

__beg	Start of range.
__end	End of range.
__a	Allocator to use (default is default allocator).

~basic_string() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::~basic_string ()  [inline], [constexpr]
```

Destroy the string instance.

basic_string() [13/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string ()  [inline], [noexcept]
```

Default constructor creates an empty string.

basic_string() [14/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            const _Alloc & __a)  [inline], [explicit]
```

Construct an empty string using allocator *a*.

basic_string() [15/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline]
```

Construct string with copy of value of *str*.

Parameters

__str	Source string.

basic_string() [16/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos,
            const _Alloc & __a = _Alloc())
```

Construct string as copy of a substring.

Parameters

__str	Source string.
__pos	Index of first character to copy from.
__a	Allocator to use.

basic_string() [17/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos,
            size_type __n)
```

Construct string as copy of a substring.

Parameters

__str	Source string.
__pos	Index of first character to copy from.
__n	Number of characters to copy.

basic_string() [18/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos,
            size_type __n,
            const _Alloc & __a)
```

Construct string as copy of a substring.

Parameters

__str	Source string.
__pos	Index of first character to copy from.
__n	Number of characters to copy.
__a	Allocator to use.

basic_string() [19/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            const _CharT * __s,
            size_type __n,
            const _Alloc & __a = _Alloc())  [inline]
```

Construct string initialized by a character array.

Parameters

_↩ _s	Source character array.
_↩ _n	Number of characters to copy.
_↩ _a	Allocator to use (default is default allocator).

NB: __s must have at least __n characters, '\0' has no special meaning.

basic_string() [20/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<typename = _RequireAllocator<_Alloc>>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            const _CharT * __s,
            const _Alloc & __a = _Alloc())  [inline]
```

Construct string as copy of a C string.

Parameters

_↩ _s	Source C string.
_↩ _a	Allocator to use (default is default allocator).

basic_string() [21/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            size_type __n,
            _CharT __c,
            const _Alloc & __a = _Alloc())  [inline]
```

Construct string as multiple characters.

Parameters

_↩ _n	Number of characters.
_↩ _c	Character to use.
_↩ _a	Allocator to use (default is default allocator).

basic_string() [22/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            basic_string< _CharT, _Traits, _Alloc > && __str)  [inline], [noexcept]
```

Move construct string.

Parameters

__str	Source string.

The newly-created string contains the exact contents of *__str*. *__str* is a valid, but unspecified string.

basic_string() [23/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            initializer_list< _CharT > __l,
            const _Alloc & __a = _Alloc())  [inline]
```

Construct string from an initializer list.

Parameters

_↩ _l	std::initializer_list of characters.
_↩ _a	Allocator to use (default is default allocator).

basic_string() [24/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            _InputIterator __beg,
            _InputIterator __end,
            const _Alloc & __a = _Alloc())  [inline]
```

Construct string as copy of a range.

Parameters

__beg	Start of range.
__end	End of range.
__a	Allocator to use (default is default allocator).

~basic_string() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::~basic_string ()  [inline], [noexcept]
```

Destroy the string instance.

5.233.3 Member Function Documentation

__resize_and_overwrite() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<typename _Operation>
void std::basic_string< _CharT, _Traits, _Alloc >::__resize_and_overwrite (
            size_type __n,
            _Operation __op)  [constexpr]
```

Non-standard version of resize_and_overwrite for C++11 and above.

__resize_and_overwrite() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<typename _Operation>
void std::basic_string< _CharT, _Traits, _Alloc >::__resize_and_overwrite (
            size_type __n,
            _Operation __op)
```

Non-standard version of resize_and_overwrite for C++11 and above.

append() [1/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::append (
            _InputIterator __first,
            _InputIterator __last)  [inline], [constexpr]
```

Append a range of characters.

Parameters

__first	Iterator referencing the first character to append.
__last	Iterator marking the end of the range.

Returns

Reference to this string.

Appends characters in the range [__first,__last) to this string.

append() [2/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::append (
            _InputIterator __first,
            _InputIterator __last)  [inline]
```

Append a range of characters.

Parameters

__first	Iterator referencing the first character to append.
__last	Iterator marking the end of the range.

Returns

Reference to this string.

Appends characters in the range [__first,__last) to this string.

append() [3/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::append (
            const _CharT * __s)  [inline], [constexpr]
```

Append a C string.

Parameters

_↩ _s	The C string to append.

Returns

Reference to this string.

append() [4/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::append (
            const _CharT * __s)  [inline]
```

Append a C string.

Parameters

_↩ _s	The C string to append.

Returns

Reference to this string.

append() [5/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::append (
            const _CharT * __s,
            size_type __n)  [inline], [constexpr]
```

Append a C substring.

Parameters

_↩ _s	The C string to append.
_↩ _n	The number of characters to append.

Returns

Reference to this string.

append() [6/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::append (
            const _CharT * __s,
            size_type __n)
```

Append a C substring.

Parameters

_↩ _s	The C string to append.
_↩ _n	The number of characters to append.

Returns

Reference to this string.

append() [7/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::append (
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline], [constexpr]
```

Append a string to this string.

Parameters

__str	The string to append.

Returns

Reference to this string.

Referenced by std::basic_string< _CharT >::append(), std::basic_string< _CharT >::append(), std::basic_string< _CharT >::append(), std::basic_string< _CharT >::append(), std::__cxx11::collate< _CharT >::do_transform(), std::basic_string< _CharT >::operator+=(), std::basic_string< _CharT >::operator+=(), std::basic_string< _CharT >::operator+=(), std::operator>>(), and resize().

append() [8/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::append (
            const basic_string< _CharT, _Traits, _Alloc > & __str)
```

Append a string to this string.

Parameters

__str	The string to append.

Returns

Reference to this string.

append() [9/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::append (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos,
            size_type __n = npos)  [inline], [constexpr]
```

Append a substring.

Parameters

__str	The string to append.
__pos	Index of the first character of str to append.
__n	The number of characters to append.

Returns

Reference to this string.

Exceptions

std::out_of_range	if *__pos* is not a valid index.

This function appends *__n* characters from *__str* starting at *__pos* to this string. If *__n* is is larger than the number of available characters in *__str*, the remainder of *__str* is appended.

append() [10/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::append (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos,
            size_type __n = npos)
```

Append a substring.

Parameters

__str	The string to append.
__pos	Index of the first character of str to append.
__n	The number of characters to append.

Returns

Reference to this string.

Exceptions

std::out_of_range	if *__pos* is not a valid index.

This function appends *__n* characters from *__str* starting at *__pos* to this string. If *__n* is is larger than the number of available characters in *__str*, the remainder of *__str* is appended.

append() [11/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::append (
            initializer_list< _CharT > __l)  [inline], [constexpr]
```

Append an initializer_list of characters.

Parameters

↩ _↩ ↩ _↩ *l*	The initializer_list of characters to append.

Returns

Reference to this string.

append() [12/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::append (
            initializer_list< _CharT > __l)  [inline]
```

Append an initializer_list of characters.

Parameters

↩ _↩ ↩ _↩ *l*	The initializer_list of characters to append.

Returns

Reference to this string.

append() [13/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::append (
            size_type __n,
            _CharT __c)  [inline], [constexpr]
```

Append multiple characters.

Parameters

_↩ _*n*	The number of characters to append.
_↩ _*c*	The character to use.

Returns

Reference to this string.

Appends __n copies of __c to this string.

append() [14/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::append (
            size_type __n,
            _CharT __c)
```

Append multiple characters.

Parameters

_↩ _n	The number of characters to append.
_↩ _c	The character to use.

Returns

Reference to this string.

Appends __n copies of __c to this string.

assign() [1/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
            _InputIterator __first,
            _InputIterator __last)  [inline], [constexpr]
```

Set value to a range of characters.

Parameters

__first	Iterator referencing the first character to append.
__last	Iterator marking the end of the range.

Returns

Reference to this string.

Sets value of string to characters in the range [__first,__last).

assign() [2/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
            _InputIterator __first,
            _InputIterator __last)  [inline]
```

Set value to a range of characters.

Parameters

__first	Iterator referencing the first character to append.
__last	Iterator marking the end of the range.

Returns

Reference to this string.

Sets value of string to characters in the range [__first,__last).

assign() [3/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
            basic_string< _CharT, _Traits, _Alloc > && __str)  [inline], [constexpr], [noexcept]
```

Set value to contents of another string.

Parameters

__str	Source string to use.

Returns

Reference to this string.

This function sets this string to the exact contents of *__str*. *__str* is a valid, but unspecified string.

assign() [4/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
            basic_string< _CharT, _Traits, _Alloc > && __str)  [inline], [noexcept]
```

Set value to contents of another string.

Parameters

__str	Source string to use.

Returns

Reference to this string.

This function sets this string to the exact contents of *__str*. *__str* is a valid, but unspecified string.

assign() [5/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
            const _CharT * __s)  [inline], [constexpr]
```

Set value to contents of a C string.

Parameters

_↩ _s	The C string to use.

Returns

Reference to this string.

This function sets the value of this string to the value of __s. The data is copied, so there is no dependence on __s once the function returns.

assign() [6/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
            const _CharT * __s)  [inline]
```

Set value to contents of a C string.

Parameters

_↩ _s	The C string to use.

Returns

Reference to this string.

This function sets the value of this string to the value of __s. The data is copied, so there is no dependence on __s once the function returns.

assign() [7/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::assign (
            const _CharT * __s,
            size_type __n)  [inline], [constexpr]
```

Set value to a C substring.

Parameters

_↩ _s	The C string to use.
_↩ _n	Number of characters to use.

Returns

Reference to this string.

This function sets the value of this string to the first __n characters of __s. If __n is is larger than the number of available characters in __s, the remainder of __s is used.

assign() [8/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
            const _CharT * __s,
            size_type __n)
```

Set value to a C substring.

Parameters

_↩ _s	The C string to use.
_↩ _n	Number of characters to use.

Returns

Reference to this string.

This function sets the value of this string to the first __n characters of __s. If __n is is larger than the number of available characters in __s, the remainder of __s is used.

assign() [9/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::assign (
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline], [constexpr]
```

Set value to contents of another string.

Parameters

__str	Source string to use.

Returns

Reference to this string.

Referenced by std::basic_string< _CharT >::assign(), std::basic_string< _CharT >::assign(), std::basic_string< _CharT >::assign(), __gnu_debug::basic_string< char >::compare(), std::basic_string< _CharT >::operator=(), std::basic_string< _CharT >::operator=(), std::basic_string< _CharT >::operator=(), std::basic_string< _CharT >::operator=(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >

assign() [10/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
            const basic_string< _CharT, _Traits, _Alloc > & __str)
```

Set value to contents of another string.

Parameters

__str	Source string to use.

Returns

Reference to this string.

assign() [11/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos,
            size_type __n = npos)  [inline], [constexpr]
```

Set value to a substring of a string.

Parameters

__str	The string to use.
__pos	Index of the first character of str.
__n	Number of characters to use.

Returns

Reference to this string.

Exceptions

std::out_of_range	if *pos* is not a valid index.

This function sets this string to the substring of *__str* consisting of *__n* characters at *__pos*. If *__n* is is larger than the number of available characters in *__str*, the remainder of *__str* is used.

assign() [12/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos,
            size_type __n = npos)  [inline]
```

Set value to a substring of a string.

Parameters

__str	The string to use.
__pos	Index of the first character of str.
__n	Number of characters to use.

Returns

Reference to this string.

Exceptions

std::out_of_range	if *pos* is not a valid index.

This function sets this string to the substring of *__str* consisting of *__n* characters at *__pos*. If *__n* is is larger than the number of available characters in *__str*, the remainder of *__str* is used.

assign() [13/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
            initializer_list< _CharT > __l)  [inline], [constexpr]
```

Set value to an initializer_list of characters.

Parameters

↩ _↩ ↩ _↩ *l*	The initializer_list of characters to assign.

Returns

Reference to this string.

assign() [14/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
            initializer_list< _CharT > __l)  [inline]
```

Set value to an initializer_list of characters.

Parameters

↩ _↩ ↩ _↩ *l*	The initializer_list of characters to assign.

Returns

Reference to this string.

assign() [15/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
            size_type __n,
            _CharT __c)  [inline], [constexpr]
```

Set value to multiple characters.

Parameters

_↩ _*n*	Length of the resulting string.
_↩ _*c*	The character to use.

Returns

Reference to this string.

This function sets the value of this string to __*n* copies of character __*c*.

assign() [16/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
            size_type __n,
            _CharT __c)  [inline]
```

Set value to multiple characters.

Parameters

_↩ _*n*	Length of the resulting string.
_↩ _*c*	The character to use.

Returns

Reference to this string.

This function sets the value of this string to __*n* copies of character __*c*.

at() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string< _CharT, _Traits, _Alloc >::at (
            size_type __n)  [inline], [nodiscard], [constexpr]
```

Provides access to the data contained in the string.

Parameters

_↩ _*n*	The index of the character to access.

Returns

Read/write reference to the character.

Exceptions

std::out_of_range	If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws out_of_range if the check fails.

at() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string< _CharT, _Traits, _Alloc >::at (
            size_type __n)  [inline]
```

Provides access to the data contained in the string.

Parameters

_↩ _n	The index of the character to access.

Returns

Read/write reference to the character.

Exceptions

std::out_of_range	If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws out_of_range if the check fails. Success results in unsharing the string.

at() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string< _CharT, _Traits, _Alloc >::at (
            size_type __n) const  [inline], [nodiscard], [constexpr]
```

Provides access to the data contained in the string.

Parameters

_↩ _n	The index of the character to access.

Returns

Read-only (const) reference to the character.

Exceptions

std::out_of_range	If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws out_of_range if the check fails.

at() [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string< _CharT, _Traits, _Alloc >::at (
            size_type __n) const  [inline]
```

Provides access to the data contained in the string.

Parameters

_↩ _n	The index of the character to access.

Returns

Read-only (const) reference to the character.

Exceptions

std::out_of_range	If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws out_of_range if the check fails.

back() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string< _CharT, _Traits, _Alloc >::back ()  [inline]
```

Returns a read/write reference to the data at the last element of the string.

back() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string< _CharT, _Traits, _Alloc >::back () const  [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns a read-only (constant) reference to the data at the last element of the string.

back() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string< _CharT, _Traits, _Alloc >::back () const  [inline], [noexcept]
```

Returns a read-only (constant) reference to the data at the last element of the string.

back() [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string< _CharT, _Traits, _Alloc >::back ()  [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read/write reference to the data at the last element of the string.

begin() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::begin ()  [inline]
```

Returns a read/write iterator that points to the first character in the string. Unshares the string.

begin() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string< _CharT, _Traits, _Alloc >::begin () const  [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns a read-only (constant) iterator that points to the first character in the string.

begin() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string< _CharT, _Traits, _Alloc >::begin () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first character in the string.

begin() [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::begin () [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read/write iterator that points to the first character in the string.

Referenced by std::basic_string< _CharT >::crend(), std::basic_string< _CharT >::erase(), std::basic_string< _CharT >::erase(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), std::regex_match(), std::regex_match(), std::regex_replace(), std::regex_replace(), std::regex_search(), std::basic_string< _CharT >::ren std::basic_string< _CharT >::rend(), std::basic_string< _CharT >::rend(), std::basic_string< _CharT >::replace(), std::basic_string< _CharT >::replace(), and std::basic_string< _CharT >::replace().

c_str() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const _CharT * std::basic_string< _CharT, _Traits, _Alloc >::c_str () const [inline], [nodiscard],
[constexpr], [noexcept]
```

Return const pointer to null-terminated contents.

This is a handle to internal data. Do not modify or dire things may happen.

Referenced by std::__cxx11::collate< _CharT >::do_compare(), std::money_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::__cxx11::collate< _CharT >::do_transform(), std::operator+(), std::operator+(), std::operator+(), std::operator+(), std::operator+(), std::regex_replace(), and std::regex_traits< _CharT >::transform_primary().

c_str() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const _CharT * std::basic_string< _CharT, _Traits, _Alloc >::c_str () const [inline], [noexcept]
```

Return const pointer to null-terminated contents.

This is a handle to internal data. Do not modify or dire things may happen.

capacity() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::capacity () const [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns the total number of characters that the string can hold before needing to allocate more memory.

Referenced by std::basic_string< _CharT >::assign(), std::basic_string< _CharT >::push_back(), reserve(), and reserve().

capacity() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::capacity () const [inline], [noexcept]
```

Returns the total number of characters that the string can hold before needing to allocate more memory.

cbegin() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string< _CharT, _Traits, _Alloc >::cbegin () const [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns a read-only (constant) iterator that points to the first character in the string.

cbegin() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string< _CharT, _Traits, _Alloc >::cbegin () const  [inline], [noexcept]
```
Returns a read-only (constant) iterator that points to the first character in the string.

cend() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string< _CharT, _Traits, _Alloc >::cend () const  [inline], [nodiscard],
[constexpr], [noexcept]
```
Returns a read-only (constant) iterator that points one past the last character in the string.

cend() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string< _CharT, _Traits, _Alloc >::cend () const  [inline], [noexcept]
```
Returns a read-only (constant) iterator that points one past the last character in the string.

clear() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::clear ()  [inline], [constexpr], [noexcept]
```
Erases the string, making it empty.
Referenced by std::__cxx11::collate< _CharT >::do_transform().

clear() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::clear ()  [inline], [noexcept]
```
Erases the string, making it empty.

compare() [1/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
            const _CharT * __s) const  [inline], [nodiscard], [constexpr], [noexcept]
```
Compare to a C string.

Parameters

_↩ _s	C string to compare against.

Returns

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before __s, 0 if their values are equivalent, or > 0 if this string is ordered after __s. Determines the effective length rlen of the strings to compare as the smallest of size() and the length of a string constructed from __s. The function then compares the two strings by calling traits::compare(data(),s,rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

compare() [2/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
            const _CharT * __s) const  [inline], [noexcept]
```

Compare to a C string.

Parameters

_↩ _s	C string to compare against.

Returns

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before __s, 0 if their values are equivalent, or > 0 if this string is ordered after __s. Determines the effective length rlen of the strings to compare as the smallest of size() and the length of a string constructed from __s. The function then compares the two strings by calling traits::compare(data(),s,rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

compare() [3/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
            const basic_string< _CharT, _Traits, _Alloc > & __str) const  [inline], [nodiscard],
[constexpr]
```

Compare to a string.

Parameters

__str	String to compare against.

Returns

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before __str, 0 if their values are equivalent, or > 0 if this string is ordered after __str. Determines the effective length rlen of the strings to compare as the smallest of size() and str.size(). The function then compares the two strings by calling traits::compare(data(), str.data(),rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

compare() [4/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
            const basic_string< _CharT, _Traits, _Alloc > & __str) const  [inline]
```

Compare to a string.

Parameters

__str	String to compare against.

Returns

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before *__str*, 0 if their values are equivalent, or > 0 if this string is ordered after *__str*. Determines the effective length rlen of the strings to compare as the smallest of size() and str.size(). The function then compares the two strings by calling traits::compare(data(), str.data(),rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

compare() [5/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
            size_type __pos,
            size_type __n,
            const basic_string< _CharT, _Traits, _Alloc > & __str) const  [inline], [nodiscard],
[constexpr]
```

Compare substring to a string.

Parameters

__pos	Index of first character of substring.
__n	Number of characters in substring.
__str	String to compare against.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n* characters starting at *__pos*. Returns an integer < 0 if the substring is ordered before *__str*, 0 if their values are equivalent, or > 0 if the substring is ordered after *__str*. Determines the effective length rlen of the strings to compare as the smallest of the length of the substring and *__str.size()*. The function then compares the two strings by calling traits::compare(substring.data(),str.data(),rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

compare() [6/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
            size_type __pos,
            size_type __n,
            const basic_string< _CharT, _Traits, _Alloc > & __str) const  [inline]
```

Compare substring to a string.

Parameters

__pos	Index of first character of substring.
__n	Number of characters in substring.
__str	String to compare against.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n* characters starting at *__pos*. Returns an integer < 0 if the substring is ordered before *__str*, 0 if their values are equivalent, or > 0 if the substring is ordered after *__str*. Determines the effective length rlen of the strings to compare as the smallest of the length of the substring and *__str.size()*. The function then compares the two strings by calling traits::compare(substring.data(),str.data(),rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

compare() [7/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
            size_type __pos,
            size_type __n1,
            const _CharT * __s) const  [inline], [nodiscard], [constexpr]
```

Compare substring to a C string.

Parameters

__pos	Index of first character of substring.
__n1	Number of characters in substring.
__s	C string to compare against.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n1* characters starting at *pos*. Returns an integer < 0 if the substring is ordered before *__s*, 0 if their values are equivalent, or > 0 if the substring is ordered after *__s*. Determines the effective length rlen of the strings to compare as the smallest of the length of the substring and the length of a string constructed from *__s*. The function then compares the two string by calling traits::compare(substring.data(),__s,rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

compare() [8/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
            size_type __pos,
            size_type __n1,
            const _CharT * __s) const  [inline]
```

Compare substring to a C string.

Parameters

__pos	Index of first character of substring.
__n1	Number of characters in substring.
__s	C string to compare against.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n1* characters starting at *pos*. Returns an integer < 0 if the substring is ordered before *__s*, 0 if their values are equivalent, or > 0 if the substring is ordered after *__s*. Determines the effective length rlen of the strings to compare as the smallest of the length of the substring and the length of a string constructed from *__s*. The function then compares the two string by calling traits::compare(substring.data(),__s,rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

compare() [9/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
            size_type __pos,
            size_type __n1,
            const _CharT * __s,
            size_type __n2) const  [inline], [nodiscard], [constexpr]
```

Compare substring against a character array.

Parameters

__pos	Index of first character of substring.
__n1	Number of characters in substring.
__s	character array to compare against.
__n2	Number of characters of s.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n1* characters starting at *__pos*. Form a string from the first *__n2* characters of *__s*. Returns an integer < 0 if this substring is ordered before the string from *__s*, 0 if their values are equivalent, or > 0 if this substring is ordered after the string from *__s*. Determines the effective length rlen of the strings to compare as the smallest of the length of the substring and *__n2*. The function then compares the two strings by calling traits↩ ::compare(substring.data(),s,rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.
NB: s must have at least n2 characters, '\0' has no special meaning.

compare() [10/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
            size_type __pos,
            size_type __n1,
            const _CharT * __s,
            size_type __n2) const  [inline]
```

Compare substring against a character array.

Parameters

__pos	Index of first character of substring.
__n1	Number of characters in substring.
__s	character array to compare against.
__n2	Number of characters of s.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the __*n1* characters starting at __*pos*. Form a string from the first __*n2* characters of __*s*. Returns an integer < 0 if this substring is ordered before the string from __*s*, 0 if their values are equivalent, or > 0 if this substring is ordered after the string from __*s*. Determines the effective length rlen of the strings to compare as the smallest of the length of the substring and __*n2*. The function then compares the two strings by calling traits::compare(substring.data(),s,rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.
NB: s must have at least n2 characters, '\0' has no special meaning.

compare() [11/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
            size_type __pos1,
            size_type __n1,
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos2,
            size_type __n2 = npos) const  [inline], [nodiscard], [constexpr]
```

Compare substring to a substring.

Parameters

__pos1	Index of first character of substring.
__n1	Number of characters in substring.
__str	String to compare against.
__pos2	Index of first character of substring of str.
__n2	Number of characters in substring of str.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the __*n1* characters starting at __*pos1*. Form the substring of __*str* from the __*n2* characters starting at __*pos2*. Returns an integer < 0 if this substring is ordered before the substring of __*str*, 0 if their values are equivalent, or > 0 if this substring is ordered after the substring of __*str*. Determines the effective length rlen of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

compare() [12/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
            size_type __pos1,
            size_type __n1,
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos2,
            size_type __n2 = npos) const  [inline]
```

Compare substring to a substring.

Parameters

__pos1	Index of first character of substring.
__n1	Number of characters in substring.
__str	String to compare against.
__pos2	Index of first character of substring of str.
__n2	Number of characters in substring of str.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n1* characters starting at *__pos1*. Form the substring of *__str* from the *__n2* characters starting at *__pos2*. Returns an integer < 0 if this substring is ordered before the substring of *__str*, 0 if their values are equivalent, or > 0 if this substring is ordered after the substring of *__str*. Determines the effective length rlen of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

copy() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::copy (
            _CharT * __s,
            size_type __n,
            size_type __pos = 0) const  [constexpr]
```

Copy substring into C string.

Parameters

__s	C string to copy value into.
__n	Number of characters to copy.
__pos	Index of first character to copy.

Returns

Number of characters actually copied

Exceptions

std::out_of_range	If __pos $>$ size().

Copies up to *__n* characters starting at *__pos* into the C string *__s*. If *__pos* is greater than size(), out_of_range is thrown.

copy() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::copy (
            _CharT * __s,
            size_type __n,
            size_type __pos = 0) const
```

Copy substring into C string.

Parameters

__s	C string to copy value into.
__n	Number of characters to copy.
__pos	Index of first character to copy.

Returns

Number of characters actually copied

Exceptions

std::out_of_range	If __pos > size().

Copies up to *__n* characters starting at *__pos* into the C string *__s*. If *__pos* is greater than size(), out_of_range is thrown.

crbegin() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::crbegin () const  [inline],
[nodiscard], [constexpr], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

crbegin() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::crbegin () const  [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

crend() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::crend () const  [inline],
[nodiscard], [constexpr], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

crend() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::crend () const  [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

data() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const _CharT * std::basic_string< _CharT, _Traits, _Alloc >::data () const  [inline], [nodiscard],
[constexpr], [noexcept]
```

Return const pointer to contents.
This is a pointer to internal data. It is undefined to modify the contents through the returned pointer. To get a pointer that allows modifying the contents use `&str[0]` instead, (or in C++17 the non-const `str.data()` overload).
Referenced by std::basic_regex< char >::assign(), std::basic_string< _CharT >::compare(), std::basic_string< _CharT >::compare(), std::basic_string< _CharT >::compare(), std::__cxx11::collate< _CharT >::do_compare(), std::__cxx11::collate< _CharT >::do_transfo std::basic_string< _CharT >::find(), std::basic_string< _CharT >::find_first_not_of(), std::basic_string< _CharT >::find_first_of(), std::basic_string< _CharT >::find_last_of(), std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _Bidirectio std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::from_bytes(), std::operator==(), std::wstring_convert< _Codecvt, _ and std::regex_traits< _CharT >::transform().

data() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const _CharT * std::basic_string< _CharT, _Traits, _Alloc >::data () const  [inline], [noexcept]
```

Return const pointer to contents.
This is a pointer to internal data. It is undefined to modify the contents through the returned pointer. To get a pointer that allows modifying the contents use `&str[0]` instead, (or in C++17 the non-const `str.data()` overload).

data() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
_CharT * std::basic_string< _CharT, _Traits, _Alloc >::data ()  [inline], [nodiscard], [constexpr],
[noexcept]
```

Return non-const pointer to contents.
This is a pointer to the character sequence held by the string. Modifying the characters in the sequence is allowed.

data() [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
_CharT * std::basic_string< _CharT, _Traits, _Alloc >::data ()  [inline]
```

Return non-const pointer to contents.
This is a pointer to the character sequence held by the string. Modifying the characters in the sequence is allowed.
The standard requires this function to be `noexcept` but for the Copy-On-Write string implementation it can throw. This function allows modifying the string contents directly, which means we must copy-on-write to unshare it, which requires allocating memory.

empty() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
bool std::basic_string< _CharT, _Traits, _Alloc >::empty () const  [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns true if the string is empty. Equivalent to `*this == ""`.
Referenced by std::basic_string< _CharT >::back(), std::basic_string< _CharT >::back(), std::basic_string< _CharT >::back(), std::basic_string< _CharT >::front(), std::basic_string< _CharT >::front(), std::basic_string< _CharT >::front(), std::tr2::operator>>(), std::basic_string< _CharT >::pop_back(), and std::basic_string< _CharT >::pop_back().

empty() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
bool std::basic_string< _CharT, _Traits, _Alloc >::empty () const  [inline], [nodiscard], [noexcept]
```

Returns true if the string is empty. Equivalent to `*this == ""`.

end() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::end ()  [inline]
```

Returns a read/write iterator that points one past the last character in the string. Unshares the string.

end() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string< _CharT, _Traits, _Alloc >::end () const  [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last character in the string.

end() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string< _CharT, _Traits, _Alloc >::end () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last character in the string.

end() [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::end ()  [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read/write iterator that points one past the last character in the string.

Referenced by std::basic_string< _CharT >::append(), std::basic_string< _CharT >::crbegin(), std::basic_string< _CharT >::erase(), std::basic_string< _CharT >::erase(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::rbegin(), std::basic_string< _CharT >::rbegin(), std::basic_string< _CharT >::rbegin(), std::regex_match(), std::regex_match(), std::regex_replace(), std::regex_replace(), std::regex_search(), std::basic_string< _CharT >::replace(), std::basic_string< _CharT >::replace(), and std::basic_string< _CharT >::r

erase() [1/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::erase (
            __const_iterator __first,
            __const_iterator __last)  [inline], [constexpr]
```

Remove a range of characters.

Parameters

__first	Iterator referencing the first character to remove.
__last	Iterator referencing the end of the range.

Returns

Iterator referencing location of first after removal.

Removes the characters in the range [first,last) from this string. The value of the string doesn't change if an error is thrown.

erase() [2/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::erase (
            __const_iterator __position)  [inline], [constexpr]
```

Remove one character.

Parameters

__position	Iterator referencing the character to remove.

Returns

iterator referencing same location after removal.

Removes the character at *__position* from this string. The value of the string doesn't change if an error is thrown.

erase() [3/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::iterator std::basic_string< _CharT, _Traits, _Alloc >↩
::erase (
            iterator __first,
            iterator __last)
```

Remove a range of characters.

Parameters

__first	Iterator referencing the first character to remove.
__last	Iterator referencing the end of the range.

Returns

Iterator referencing location of first after removal.

Removes the characters in the range [first,last) from this string. The value of the string doesn't change if an error is thrown.

erase() [4/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::erase (
            iterator __position)  [inline]
```

Remove one character.

Parameters

__position	Iterator referencing the character to remove.

Returns

iterator referencing same location after removal.

Removes the character at *__position* from this string. The value of the string doesn't change if an error is thrown.

erase() [5/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::erase (
            size_type __pos = 0,
            size_type __n = npos)  [inline], [constexpr]
```

Remove characters.

Parameters

__pos	Index of first character to remove (default 0).
__n	Number of characters to remove (default remainder).

Returns

Reference to this string.

Exceptions

std::out_of_range	If *pos* is beyond the end of this string.

Removes *__n* characters from this string starting at *__pos*. The length of the string is reduced by *__n*. If there are < *__n* characters to remove, the remainder of the string is truncated. If *__p* is beyond end of string, out_of_range is thrown. The value of the string doesn't change if an error is thrown.
Referenced by std::getline(), std::operator>>(), std::basic_string< _CharT >::pop_back(), and resize().

erase() [6/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::erase (
            size_type __pos = 0,
            size_type __n = npos)  [inline]
```

Remove characters.

Parameters

__pos	Index of first character to remove (default 0).
__n	Number of characters to remove (default remainder).

Returns

Reference to this string.

Exceptions

std::out_of_range	If *pos* is beyond the end of this string.

Removes *__n* characters from this string starting at *__pos*. The length of the string is reduced by *__n*. If there are < *__n* characters to remove, the remainder of the string is truncated. If *__p* is beyond end of string, out_of_range is thrown. The value of the string doesn't change if an error is thrown.

find() [1/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find (
            _CharT __c,
            size_type __pos = 0) const  [nodiscard], [constexpr], [noexcept]
```

Find position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for *__c* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and std::size().

find() [2/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find (
            _CharT __c,
            size_type __pos = 0) const  [noexcept]
```

Find position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for *__c* within this string. If found, returns the index where it was found. If not found, returns npos.

find() [3/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [nodiscard], [constexpr], [noexcept]
```

Find position of a C substring.

Parameters

__s	C string to locate.
__pos	Index of character to search from.
__n	Number of characters from *s* to search for.

Returns

Index of start of first occurrence.

Starting from __pos, searches forward for the first __n characters in __s within this string. If found, returns the index where it begins. If not found, returns npos.
References std::data(), npos, and std::size().
Referenced by std::basic_string< _CharT >::find(), std::basic_string< _CharT >::find(), std::basic_string< _CharT >::find(), and std::basic_string< _CharT >::find_first_of().

find() [4/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [noexcept]
```

Find position of a C substring.

Parameters

__s	C string to locate.
__pos	Index of character to search from.
__n	Number of characters from *s* to search for.

Returns

Index of start of first occurrence.

Starting from __pos, searches forward for the first __n characters in __s within this string. If found, returns the index where it begins. If not found, returns npos.

find() [5/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find (
            const _CharT * __s,
            size_type __pos = 0) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a C string.

Parameters

__s	C string to locate.
__pos	Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from __pos, searches forward for the value of __s within this string. If found, returns the index where it begins. If not found, returns npos.

find() [6/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find (
            const _CharT * __s,
            size_type __pos = 0) const  [inline], [noexcept]
```

Find position of a C string.

Parameters

__s	C string to locate.
__pos	Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from *__pos*, searches forward for the value of *__s* within this string. If found, returns the index where it begins. If not found, returns npos.

find() [7/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos = 0) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a string.

Parameters

__str	String to locate.
__pos	Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from *__pos*, searches forward for value of *__str* within this string. If found, returns the index where it begins. If not found, returns npos.

find() [8/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos = 0) const  [inline], [noexcept]
```

Find position of a string.

Parameters

__str	String to locate.
__pos	Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from *__pos*, searches forward for value of *__str* within this string. If found, returns the index where it begins. If not found, returns npos.

find_first_not_of() [1/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_first_not_of (
            _CharT __c,
            size_type __pos = 0) const  [nodiscard], [constexpr], [noexcept]
```

Find position of a different character.

Parameters

__c	Character to avoid.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character other than *__c* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and std::size().

find_first_not_of() [2/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of (
            _CharT __c,
            size_type __pos = 0) const  [noexcept]
```

Find position of a different character.

Parameters

__c	Character to avoid.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character other than *__c* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_not_of() [3/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_first_not_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [nodiscard], [constexpr], [noexcept]
```

Find position of a character not in C substring.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search from.
__n	Number of characters from __s to consider.

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character not contained in the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and std::size().

find_first_not_of() [4/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [noexcept]
```

Find position of a character not in C substring.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search from.
__n	Number of characters from __s to consider.

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character not contained in the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_not_of() [5/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of (
            const _CharT * __s,
            size_type __pos = 0) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a character not in C string.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character not contained in *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_not_of() [6/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of (
            const _CharT * __s,
            size_type __pos = 0) const  [inline], [noexcept]
```

Find position of a character not in C string.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character not contained in *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_not_of() [7/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos = 0) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a character not in string.

Parameters

__str	String containing characters to avoid.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character not contained in *__str* within this string. If found, returns the index where it was found. If not found, returns npos.
Referenced by std::basic_string< _CharT >::find_first_not_of(), std::basic_string< _CharT >::find_first_not_of(), and std::basic_string< _CharT >::find_first_not_of().

find_first_not_of() [8/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos = 0) const  [inline], [noexcept]
```

Find position of a character not in string.

Parameters

__str	String containing characters to avoid.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character not contained in *__str* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_of() [1/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of (
            _CharT __c,
            size_type __pos = 0) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for the character *__c* within this string. If found, returns the index where it was found. If not found, returns npos.
Note: equivalent to find(__c, __pos).

find_first_of() [2/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of (
            _CharT __c,
            size_type __pos = 0) const  [inline], [noexcept]
```

Find position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for the character *__c* within this string. If found, returns the index where it was found. If not found, returns npos.
Note: equivalent to find(__c, __pos).

find_first_of() [3/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_first_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [nodiscard], [constexpr], [noexcept]
```

Find position of a character of C substring.

Parameters

__s	String containing characters to locate.
__pos	Index of character to search from.
__n	Number of characters from s to search for.

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for one of the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and std::size().

find_first_of() [4/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [noexcept]
```

Find position of a character of C substring.

Parameters

__s	String containing characters to locate.
__pos	Index of character to search from.
__n	Number of characters from s to search for.

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for one of the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_of() [5/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of (
            const _CharT * __s,
            size_type __pos = 0) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a character of C string.

Parameters

__s	String containing characters to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for one of the characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_of() [6/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of (
            const _CharT * __s,
            size_type __pos = 0) const  [inline], [noexcept]
```

Find position of a character of C string.

Parameters

__s	String containing characters to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for one of the characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_of() [7/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos = 0) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a character of string.

Parameters

__str	String containing characters to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for one of the characters of *__str* within this string. If found, returns the index where it was found. If not found, returns npos.
Referenced by std::basic_string< _CharT >::find_first_of(), and std::basic_string< _CharT >::find_first_of().

find_first_of() [8/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos = 0) const  [inline], [noexcept]
```

Find position of a character of string.

Parameters

__str	String containing characters to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for one of the characters of *__str* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_not_of() [1/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_last_not_of (
            _CharT __c,
            size_type __pos = npos) const  [nodiscard], [constexpr], [noexcept]
```

Find last position of a different character.

Parameters

__c	Character to avoid.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character other than *__c* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and std::size().

find_last_not_of() [2/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (
            _CharT __c,
            size_type __pos = npos) const  [noexcept]
```

Find last position of a different character.

Parameters

__c	Character to avoid.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character other than *__c* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_not_of() [3/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_last_not_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [nodiscard], [constexpr], [noexcept]
```

Find last position of a character not in C substring.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search back from.
__n	Number of characters from s to consider.

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character not contained in the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and std::size().

find_last_not_of() [4/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [noexcept]
```

Find last position of a character not in C substring.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search back from.
__n	Number of characters from s to consider.

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character not contained in the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_not_of() [5/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (
            const _CharT * __s,
            size_type __pos = npos) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a character not in C string.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character not contained in *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_not_of() [6/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (
            const _CharT * __s,
            size_type __pos = npos) const  [inline], [noexcept]
```

Find last position of a character not in C string.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character not contained in *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_not_of() [7/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos = npos) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a character not in string.

Parameters

__str	String containing characters to avoid.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character not contained in *__str* within this string. If found, returns the index where it was found. If not found, returns npos.
Referenced by std::basic_string< _CharT >::find_last_not_of(), std::basic_string< _CharT >::find_last_not_of(), and std::basic_string< _CharT >::find_last_not_of().

find_last_not_of() [8/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos = npos) const  [inline], [noexcept]
```

Find last position of a character not in string.

Parameters

__str	String containing characters to avoid.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character not contained in *__str* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_of() [1/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of (
            _CharT __c,
            size_type __pos = npos) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for *__c* within this string. If found, returns the index where it was found. If not found, returns npos.
Note: equivalent to rfind(__c, __pos).

find_last_of() [2/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of (
            _CharT __c,
            size_type __pos = npos) const  [inline], [noexcept]
```

Find last position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for *__c* within this string. If found, returns the index where it was found. If not found, returns npos.
Note: equivalent to rfind(__c, __pos).

find_last_of() [3/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_last_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [nodiscard], [constexpr], [noexcept]
```

Find last position of a character of C substring.

Parameters

__s	C string containing characters to locate.
__pos	Index of character to search back from.
__n	Number of characters from s to search for.

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for one of the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and std::size().

find_last_of() [4/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [noexcept]
```

Find last position of a character of C substring.

Parameters

__s	C string containing characters to locate.
__pos	Index of character to search back from.
__n	Number of characters from s to search for.

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for one of the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_of() [5/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of (
            const _CharT * __s,
            size_type __pos = npos) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a character of C string.

Parameters

__s	C string containing characters to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for one of the characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_of() [6/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of (
            const _CharT * __s,
            size_type __pos = npos) const  [inline], [noexcept]
```

Find last position of a character of C string.

Parameters

__s	C string containing characters to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for one of the characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_of() [7/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos = npos) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a character of string.

Parameters

__str	String containing characters to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for one of the characters of *__str* within this string. If found, returns the index where it was found. If not found, returns npos.
Referenced by std::basic_string< _CharT >::find_last_of(), and std::basic_string< _CharT >::find_last_of().

find_last_of() [8/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos = npos) const  [inline], [noexcept]
```

Find last position of a character of string.

Parameters

__str	String containing characters to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from __*pos*, searches backward for one of the characters of __*str* within this string. If found, returns the index where it was found. If not found, returns npos.

front() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string< _CharT, _Traits, _Alloc >::front ()  [inline]
```

Returns a read/write reference to the data at the first element of the string.

front() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string< _CharT, _Traits, _Alloc >::front () const  [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns a read-only (constant) reference to the data at the first element of the string.

front() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string< _CharT, _Traits, _Alloc >::front () const  [inline], [noexcept]
```

Returns a read-only (constant) reference to the data at the first element of the string.

front() [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string< _CharT, _Traits, _Alloc >::front ()  [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read/write reference to the data at the first element of the string.

get_allocator() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
allocator_type std::basic_string< _CharT, _Traits, _Alloc >::get_allocator () const  [inline],
[nodiscard], [constexpr], [noexcept]
```

Return copy of allocator used to construct this string.

Referenced by std::basic_string< _CharT >::~basic_string(), std::basic_string< _CharT >::assign(), std::basic_string< _CharT >::clear(), std::operator+(), std::operator+(), std::operator+(), std::operator+(), std::operator+(), reserve(), reserve(), and swap().

get_allocator() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
allocator_type std::basic_string< _CharT, _Traits, _Alloc >::get_allocator () const  [inline],
[noexcept]
```

Return copy of allocator used to construct this string.

insert() [1/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::insert (
            __const_iterator __p,
            _CharT __c)  [inline], [constexpr]
```

Insert one character.

Parameters

__p	Iterator referencing position in string to insert at.
__c	The character to insert.

Returns

Iterator referencing newly inserted char.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts character *__c* at position referenced by *__p*. If adding character causes the length to exceed max_size(), length_error is thrown. If *__p* is beyond end of string, out_of_range is thrown. The value of the string doesn't change if an error is thrown.

insert() [2/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
iterator std::basic_string< _CharT, _Traits, _Alloc >::insert (
            const_iterator __p,
            _InputIterator __beg,
            _InputIterator __end)  [inline], [constexpr]
```

Insert a range of characters.

Parameters

__p	Const_iterator referencing location in string to insert at.
__beg	Start of range.
__end	End of range.

Returns

Iterator referencing the first inserted char.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts characters in range [beg,end). If adding characters causes the length to exceed max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

insert() [3/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::insert (
            const_iterator __p,
            initializer_list< _CharT > __l)  [inline], [constexpr]
```

Insert an initializer_list of characters.

Parameters

_↩ _p	Iterator referencing location in string to insert at.
_↩ _l	The initializer_list of characters to insert.

Exceptions

std::length_error	If new length exceeds `max_size()`.

insert() [4/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::insert (
            const_iterator __p,
            size_type __n,
            _CharT __c)  [inline], [constexpr]
```

Insert multiple characters.

Parameters

_↩ _p	Const_iterator referencing location in string to insert at.
_↩ _n	Number of characters to insert
_↩ _c	The character to insert.

Returns

Iterator referencing the first inserted char.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts *__n* copies of character *__c* starting at the position referenced by iterator *__p*. If adding characters causes the length to exceed max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.
Referenced by std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), and std::basic_string< _CharT >::insert().

insert() [5/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::insert (
            iterator __p,
            _CharT __c)  [inline]
```

Insert one character.

Parameters

__p	Iterator referencing position in string to insert at.
__c	The character to insert.

Returns

Iterator referencing newly inserted char.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts character *__c* at position referenced by *__p*. If adding character causes the length to exceed max_size(), length_error is thrown. If *__p* is beyond end of string, out_of_range is thrown. The value of the string doesn't change if an error is thrown.

insert() [6/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator>
void std::basic_string< _CharT, _Traits, _Alloc >::insert (
             iterator __p,
             _InputIterator __beg,
             _InputIterator __end)  [inline]
```

Insert a range of characters.

Parameters

__p	Iterator referencing location in string to insert at.
__beg	Start of range.
__end	End of range.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts characters in range [__beg,__end). If adding characters causes the length to exceed max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

insert() [7/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::insert (
             iterator __p,
             initializer_list< _CharT > __l)  [inline]
```

Insert an initializer_list of characters.

Parameters

_↩ _p	Iterator referencing location in string to insert at.
_↩ _l	The initializer_list of characters to insert.

Exceptions

std::length_error	If new length exceeds `max_size()`.

insert() [8/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::insert (
            iterator __p,
            size_type __n,
            _CharT __c)  [inline]
```

Insert multiple characters.

Parameters

_↩ _p	Iterator referencing location in string to insert at.
_↩ _n	Number of characters to insert
_↩ _c	The character to insert.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts *__n* copies of character *__c* starting at the position referenced by iterator *__p*. If adding characters causes the length to exceed max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

insert() [9/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::insert (
            size_type __pos,
            const _CharT * __s)  [inline], [constexpr]
```

Insert a C string.

Parameters

__pos	Position in string to insert at.
__s	The C string to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.
std::out_of_range	If *pos* is beyond the end of this string.

Inserts the first *n* characters of *__s* starting at *__pos*. If adding characters causes the length to exceed max_size(), length_error is thrown. If *__pos* is beyond end(), out_of_range is thrown. The value of the string doesn't change if an error is thrown.

insert() [10/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::insert (
            size_type __pos,
            const _CharT * __s)  [inline]
```

Insert a C string.

Parameters

__pos	Position in string to insert at.
__s	The C string to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.
std::out_of_range	If *pos* is beyond the end of this string.

Inserts the first *n* characters of *__s* starting at *__pos*. If adding characters causes the length to exceed max_size(), length_error is thrown. If *__pos* is beyond end(), out_of_range is thrown. The value of the string doesn't change if an error is thrown.

insert() [11/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::insert (
            size_type __pos,
            const _CharT * __s,
            size_type __n)  [inline], [constexpr]
```

Insert a C substring.

Parameters

__pos	Position in string to insert at.
__s	The C string to insert.
__n	The number of characters to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.
std::out_of_range	If *__pos* is beyond the end of this string.

Inserts the first *__n* characters of *__s* starting at *__pos*. If adding characters causes the length to exceed max_size(), length_error is thrown. If *__pos* is beyond end(), out_of_range is thrown. The value of the string doesn't change if an error is thrown.

insert() [12/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::insert (
            size_type __pos,
            const _CharT * __s,
            size_type __n)
```

Insert a C substring.

Parameters

__pos	Position in string to insert at.
__s	The C string to insert.
__n	The number of characters to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.
std::out_of_range	If *__pos* is beyond the end of this string.

Inserts the first *__n* characters of *__s* starting at *__pos*. If adding characters causes the length to exceed max_size(), length_error is thrown. If *__pos* is beyond end(), out_of_range is thrown. The value of the string doesn't change if an error is thrown.

insert() [13/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::insert (
            size_type __pos,
            size_type __n,
            _CharT __c)  [inline], [constexpr]
```

Insert multiple characters.

Parameters

__pos	Index in string to insert at.
__n	Number of characters to insert
__c	The character to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.
std::out_of_range	If *__pos* is beyond the end of this string.

Inserts *__n* copies of character *__c* starting at index *__pos*. If adding characters causes the length to exceed max_↩ size(), length_error is thrown. If *__pos* > length(), out_of_range is thrown. The value of the string doesn't change if an error is thrown.

insert() [14/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::insert (
            size_type __pos,
            size_type __n,
            _CharT __c)  [inline]
```

Insert multiple characters.

Parameters

__pos	Index in string to insert at.
__n	Number of characters to insert
__c	The character to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.
std::out_of_range	If *__pos* is beyond the end of this string.

Inserts *__n* copies of character *__c* starting at index *__pos*. If adding characters causes the length to exceed max_↩ size(), length_error is thrown. If *__pos* > length(), out_of_range is thrown. The value of the string doesn't change if an error is thrown.

insert() [15/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::insert (
            size_type __pos1,
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline], [constexpr]
```

Insert value of a string.

Parameters

__pos1	Position in string to insert at.
__str	The string to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts value of *__str* starting at *__pos1*. If adding characters causes the length to exceed max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

insert() [16/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::insert (
            size_type __pos1,
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline]
```

Insert value of a string.

Parameters

__pos1	Position in string to insert at.
__str	The string to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts value of *__str* starting at *__pos1*. If adding characters causes the length to exceed max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

insert() [17/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::insert (
            size_type __pos1,
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos2,
            size_type __n = npos)  [inline], [constexpr]
```

Insert a substring.

Parameters

__pos1	Position in string to insert at.
__str	The string to insert.
__pos2	Start of characters in str to insert.
__n	Number of characters to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.
std::out_of_range	If *pos1* > size() or *__pos2* > *str.size()*.

Starting at *pos1*, insert *__n* character of *__str* beginning with *__pos2*. If adding characters causes the length to exceed max_size(), length_error is thrown. If *__pos1* is beyond the end of this string or *__pos2* is beyond the end of *__str*, out_of_range is thrown. The value of the string doesn't change if an error is thrown.

insert() [18/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::insert (
            size_type __pos1,
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos2,
            size_type __n = npos)  [inline]
```

Insert a substring.

Parameters

__pos1	Position in string to insert at.
__str	The string to insert.
__pos2	Start of characters in str to insert.
__n	Number of characters to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.
std::out_of_range	If *pos1* > size() or *__pos2* > *str.size()*.

Starting at *pos1*, insert *__n* character of *__str* beginning with *__pos2*. If adding characters causes the length to exceed max_size(), length_error is thrown. If *__pos1* is beyond the end of this string or *__pos2* is beyond the end of *__str*, out_of_range is thrown. The value of the string doesn't change if an error is thrown.

length() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::length () const  [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns the number of characters in the string, not including any null-termination.

Referenced by std::basic_string< _CharT >::basic_string(), std::basic_string< _CharT >::basic_string(), std::__cxx11::collate< _CharT std::__cxx11::collate< _CharT >::do_transform(), std::regex_replace(), and reserve().

length() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::length () const  [inline], [noexcept]
```

Returns the number of characters in the string, not including any null-termination.

max_size() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::max_size () const  [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns the size() of the largest possible string.

Referenced by std::getline(), and std::operator>>().

max_size() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::max_size () const  [inline], [noexcept]
```

Returns the size() of the largest possible string.

operator+=() [1/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator+= (
            _CharT __c)  [inline], [constexpr]
```

Append a character.

Parameters

_↩ _c	The character to append.

Returns

Reference to this string.

operator+=() [2/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator+= (
            _CharT __c)  [inline]
```

Append a character.

Parameters

_↩ _c	The character to append.

Returns

Reference to this string.

operator+=() [3/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator+= (
            const _CharT * __s)  [inline], [constexpr]
```

Append a C string.

Parameters

_↩ _s	The C string to append.

Returns

Reference to this string.

operator+=() [4/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator+= (
            const _CharT * __s)  [inline]
```

Append a C string.

Parameters

_↩ _s	The C string to append.

Returns

Reference to this string.

operator+=() [5/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator+= (
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline], [constexpr]
```

Append a string to this string.

Parameters

__str	The string to append.

Returns

Reference to this string.

operator+=() [6/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator+= (
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline]
```

Append a string to this string.

Parameters

__str	The string to append.

Returns

Reference to this string.

operator+=() [7/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator+= (
            initializer_list< _CharT > __l)  [inline], [constexpr]
```

Append an initializer_list of characters.

Parameters

↩ _↩ ↩ _↩ *l*	The initializer_list of characters to be appended.

Returns

Reference to this string.

operator+=() [8/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator+= (
            initializer_list< _CharT > __l)  [inline]
```

Append an initializer_list of characters.

Parameters

↩ _↩ ↩ _↩ *l*	The initializer_list of characters to be appended.

Returns

Reference to this string.

operator=() [1/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator= (
            _CharT __c)  [inline], [constexpr]
```

Set value to string of length 1.

Parameters

_↩ _c	Source character.

Assigning to a character makes this string length 1 and (*this)[0] == *c*.

operator=() [2/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator= (
            _CharT __c)  [inline]
```

Set value to string of length 1.

Parameters

_↩ _c	Source character.

Assigning to a character makes this string length 1 and (*this)[0] == *c*.

operator=() [3/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator= (
            basic_string< _CharT, _Traits, _Alloc > && __str)  [inline], [noexcept]
```

Move assign the value of *str* to this string.

Parameters

__str	Source string.

The contents of *str* are moved into this string (without copying). *str* is a valid, but unspecified string.

operator=() [4/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator= (
            basic_string< _CharT, _Traits, _Alloc > && __str)  [inline], [constexpr], [noexcept]
```

Move assign the value of *str* to this string.

Parameters

__str	Source string.

The contents of *str* are moved into this string (without copying). *str* is a valid, but unspecified string.

operator=() [5/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator= (
            const _CharT * __s)  [inline], [constexpr]
```

Copy contents of *s* into this string.

Parameters

_↩ _s	Source null-terminated string.

operator=() [6/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator= (
            const _CharT * __s)  [inline]
```

Copy contents of *s* into this string.

Parameters

_↩ _s	Source null-terminated string.

operator=() [7/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator= (
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline], [constexpr]
```

Assign the value of *str* to this string.

Parameters

__str	Source string.

operator=() [8/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator= (
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline]
```

Assign the value of *str* to this string.

Parameters

__str	Source string.

operator=() [9/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator= (
            initializer_list< _CharT > __l)  [inline], [constexpr]
```

Set value to string constructed from initializer list.

Parameters

↩ _↩ ↩ _↩ *l*	std::initializer_list.

operator=() [10/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator= (
            initializer_list< _CharT > __l)  [inline]
```

Set value to string constructed from initializer list.

Parameters

↩ _↩ ↩ _↩ *l*	std::initializer_list.

operator[]() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string< _CharT, _Traits, _Alloc >::operator[] (
            size_type __pos)  [inline], [nodiscard], [constexpr]
```

Subscript access to the data contained in the string.

Parameters

__pos	The index of the character to access.

Returns

Read/write reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out_of_range lookups are not defined. (For checked lookups see at().)

operator[]() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string< _CharT, _Traits, _Alloc >::operator[] (
            size_type __pos)  [inline]
```

Subscript access to the data contained in the string.

Parameters

__pos	The index of the character to access.

Returns

Read/write reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out_of_range lookups are not defined. (For checked lookups see at().) Unshares the string.

operator[]() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string< _CharT, _Traits, _Alloc >::operator[] (
            size_type __pos) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Subscript access to the data contained in the string.

Parameters

__pos	The index of the character to access.

Returns

Read-only (constant) reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out_of_range lookups are not defined. (For checked lookups see at().)
Referenced by std::basic_string< _CharT >::back(), std::basic_string< _CharT >::back(), std::basic_string< _CharT >::back(), std::basic_string< _CharT >::front(), std::basic_string< _CharT >::front(), and std::basic_string< _CharT >::front().

operator[]() [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string< _CharT, _Traits, _Alloc >::operator[] (
            size_type __pos) const  [inline], [noexcept]
```

Subscript access to the data contained in the string.

Parameters

__pos	The index of the character to access.

Returns

Read-only (constant) reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out_of_range lookups are not defined. (For checked lookups see at().)

pop_back() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::pop_back ()  [inline]
```

Remove the last character.
The string must be non-empty.

pop_back() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::pop_back ()  [inline], [constexpr], [noexcept]
```

Remove the last character.
The string must be non-empty.

push_back() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::push_back (
            _CharT __c)  [inline], [constexpr]
```

Append a single character.

Parameters

_↩ _c	Character to append.

Referenced by std::__cxx11::collate< _CharT >::do_transform(), std::basic_string< _CharT >::operator+=(), std::tr2::operator>>(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow().

push_back() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::push_back (
            _CharT __c)  [inline]
```

Append a single character.

Parameters

_↩ _c	Character to append.

rbegin() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::rbegin ()  [inline]
```

Returns a read/write reverse iterator that points to the last character in the string. Iteration is done in reverse element order. Unshares the string.

rbegin() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::rbegin () const  [inline],
[nodiscard], [constexpr], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

rbegin() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::rbegin () const  [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

rbegin() [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::rbegin ()  [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns a read/write reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

rend() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::rend ()  [inline]
```

Returns a read/write reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order. Unshares the string.

rend() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::rend () const  [inline],
[nodiscard], [constexpr], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

rend() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::rend () const  [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

rend() [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::rend ()  [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns a read/write reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

replace() [1/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            __const_iterator __i1,
            __const_iterator __i2,
            const _CharT * __s)  [inline], [constexpr]
```

Replace range of characters with C string.

Parameters

_↩ _i1	Iterator referencing start of range to replace.
_↩ _i2	Iterator referencing end of range to replace.
_↩ _s	C string value to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, the characters of __*s* are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [2/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            __const_iterator __i1,
            __const_iterator __i2,
            const _CharT * __s,
            size_type __n)  [inline], [constexpr]
```

Replace range of characters with C substring.

Parameters

_↩ _*i1*	Iterator referencing start of range to replace.
_↩ _*i2*	Iterator referencing end of range to replace.
_↩ _*s*	C string value to insert.
_↩ _*n*	Number of characters from s to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, the first __*n* characters of __*s* are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [3/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            __const_iterator __i1,
            __const_iterator __i2,
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline], [constexpr]
```

Replace range of characters with string.

Parameters

__i1	Iterator referencing start of range to replace.
__i2	Iterator referencing end of range to replace.
__str	String value to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, the value of *__str* is inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [4/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            __const_iterator __i1,
            __const_iterator __i2,
            size_type __n,
            _CharT __c)  [inline], [constexpr]
```

Replace range of characters with multiple characters.

Parameters

_↩ _i1	Iterator referencing start of range to replace.
_↩ _i2	Iterator referencing end of range to replace.
_↩ _n	Number of characters to insert.
_↩ _c	Character to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, *__n* copies of *__c* are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [5/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            const_iterator __i1,
            const_iterator __i2,
            _InputIterator __k1,
            _InputIterator __k2)  [inline], [constexpr]
```

Replace range of characters with range.

Parameters

__i1	Iterator referencing start of range to replace.
__i2	Iterator referencing end of range to replace.
__k1	Iterator referencing start of range to insert.
__k2	Iterator referencing end of range to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, characters in the range [__k1,__k2) are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [6/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            const_iterator __i1,
            const_iterator __i2,
            initializer_list< _CharT > __l)  [inline], [constexpr]
```

Replace range of characters with initializer_list.

Parameters

_↩ _i1	Iterator referencing start of range to replace.
_↩ _i2	Iterator referencing end of range to replace.
_↩ _l	The initializer_list of characters to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, characters in the range [__k1,__k2) are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [7/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            iterator __i1,
            iterator __i2,
            _InputIterator __k1,
            _InputIterator __k2)  [inline]
```

Replace range of characters with range.

Parameters

__i1	Iterator referencing start of range to replace.
__i2	Iterator referencing end of range to replace.
__k1	Iterator referencing start of range to insert.
__k2	Iterator referencing end of range to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, characters in the range [__k1,__k2) are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [8/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            iterator __i1,
            iterator __i2,
            const _CharT * __s)  [inline]
```

Replace range of characters with C string.

Parameters

_↩ _i1	Iterator referencing start of range to replace.
_↩ _i2	Iterator referencing end of range to replace.
_↩ _s	C string value to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, the characters of __*s* are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [9/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            iterator __i1,
            iterator __i2,
            const _CharT * __s,
            size_type __n)  [inline]
```

Replace range of characters with C substring.

Parameters

_↩ _*i1*	Iterator referencing start of range to replace.
_↩ _*i2*	Iterator referencing end of range to replace.
_↩ _*s*	C string value to insert.
_↩ _*n*	Number of characters from s to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, the first __*n* characters of __*s* are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [10/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            iterator __i1,
            iterator __i2,
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline]
```

Replace range of characters with string.

Parameters

__i1	Iterator referencing start of range to replace.
__i2	Iterator referencing end of range to replace.
__str	String value to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, the value of *__str* is inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [11/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            iterator __i1,
            iterator __i2,
            initializer_list< _CharT > __l)  [inline]
```

Replace range of characters with initializer_list.

Parameters

_↩ _i1	Iterator referencing start of range to replace.
_↩ _i2	Iterator referencing end of range to replace.
_↩ _l	The initializer_list of characters to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, characters in the range [__k1,__k2) are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [12/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            iterator __i1,
```

```
            iterator __i2,
            size_type __n,
            _CharT __c)  [inline]
```

Replace range of characters with multiple characters.

Parameters

_↩ _i1	Iterator referencing start of range to replace.
_↩ _i2	Iterator referencing end of range to replace.
_↩ _n	Number of characters to insert.
_↩ _c	Character to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, __*n* copies of __*c* are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [13/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            size_type __pos,
            size_type __n,
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline], [constexpr]
```

Replace characters with value from another string.

Parameters

__pos	Index of first character to replace.
__n	Number of characters to be replaced.
__str	String to insert.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *pos* is beyond the end of this string.
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__pos,__pos+__n) from this string. In place, the value of __*str* is inserted. If __*pos* is beyond end of string, out_of_range is thrown. If the length of the result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.
Referenced by std::basic_string< _CharT >::append(), std::basic_string< _CharT >::assign(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::replace(), std::basic_string< _CharT >::replace(), std::basic_string< _CharT >::replace(), std::basic_string< _CharT >::replace(), std::basic_string< _CharT >::replace(), std::basic_string< _CharT >::replace(), std::basic_string< _CharT >::replace(), std::basic_string< _CharT >::replace(), std::basic_string< _CharT >::replace(), and std::basic_string< _CharT >::replace().

replace() [14/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            size_type __pos,
            size_type __n,
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline]
```

Replace characters with value from another string.

Parameters

__pos	Index of first character to replace.
__n	Number of characters to be replaced.
__str	String to insert.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *pos* is beyond the end of this string.
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__pos,__pos+__n) from this string. In place, the value of __*str* is inserted. If __*pos* is beyond end of string, out_of_range is thrown. If the length of the result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [15/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            size_type __pos,
            size_type __n1,
            const _CharT * __s)  [inline], [constexpr]
```

Replace characters with value of a C string.

Parameters

__pos	Index of first character to replace.
__n1	Number of characters to be replaced.
__s	C string to insert.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *pos* > size().
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__pos,__pos + __n1) from this string. In place, the characters of *__s* are inserted. If *__pos* is beyond end of string, out_of_range is thrown. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [16/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            size_type __pos,
            size_type __n1,
            const _CharT * __s)  [inline]
```

Replace characters with value of a C string.

Parameters

__pos	Index of first character to replace.
__n1	Number of characters to be replaced.
__s	C string to insert.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *pos* > size().
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__pos,__pos + __n1) from this string. In place, the characters of *__s* are inserted. If *__pos* is beyond end of string, out_of_range is thrown. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [17/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::replace
(
            size_type __pos,
            size_type __n1,
            const _CharT * __s,
            size_type __n2)  [inline], [constexpr]
```

Replace characters with value of a C substring.

Parameters

__pos	Index of first character to replace.
__n1	Number of characters to be replaced.
__s	C string to insert.
__n2	Number of characters from *s* to use.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *pos1* > size().
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__pos,__pos + __n1) from this string. In place, the first *__n2* characters of *__s* are inserted, or all of *__s* if *__n2* is too large. If *__pos* is beyond end of string, out_of_range is thrown. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [18/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            size_type __pos,
            size_type __n1,
            const _CharT * __s,
            size_type __n2)
```

Replace characters with value of a C substring.

Parameters

__pos	Index of first character to replace.
__n1	Number of characters to be replaced.
__s	C string to insert.
__n2	Number of characters from *s* to use.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *pos1* > size().
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__pos,__pos + __n1) from this string. In place, the first *__n2* characters of *__s* are inserted, or all of *__s* if *__n2* is too large. If *__pos* is beyond end of string, out_of_range is thrown. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [19/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            size_type __pos,
            size_type __n1,
            size_type __n2,
            _CharT __c)  [inline], [constexpr]
```

Replace characters with multiple characters.

Parameters

__pos	Index of first character to replace.
__n1	Number of characters to be replaced.
__n2	Number of characters to insert.
__c	Character to insert.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *__pos* > size().
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [pos,pos + n1) from this string. In place, *__n2* copies of *__c* are inserted. If *__pos* is beyond end of string, out_of_range is thrown. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [20/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            size_type __pos,
            size_type __n1,
            size_type __n2,
            _CharT __c)  [inline]
```

Replace characters with multiple characters.

Parameters

__pos	Index of first character to replace.

Parameters

__n1	Number of characters to be replaced.
__n2	Number of characters to insert.
__c	Character to insert.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *__pos* > size().
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [pos,pos + n1) from this string. In place, *__n2* copies of *__c* are inserted. If *__pos* is beyond end of string, out_of_range is thrown. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [21/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            size_type __pos1,
            size_type __n1,
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos2,
            size_type __n2 = npos)  [inline], [constexpr]
```

Replace characters with value from another string.

Parameters

__pos1	Index of first character to replace.
__n1	Number of characters to be replaced.
__str	String to insert.
__pos2	Index of first character of str to use.
__n2	Number of characters from str to use.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *__pos1* > size() or *__pos2* > __str.size().
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__pos1,__pos1 + n) from this string. In place, the value of *__str* is inserted. If *__pos* is beyond end of string, out_of_range is thrown. If the length of the result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [22/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            size_type __pos1,
            size_type __n1,
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos2,
            size_type __n2 = npos)  [inline]
```

Replace characters with value from another string.

Parameters

__pos1	Index of first character to replace.
__n1	Number of characters to be replaced.
__str	String to insert.
__pos2	Index of first character of str to use.
__n2	Number of characters from str to use.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *__pos1* > size() or *__pos2* > __str.size().
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__pos1,__pos1 + n) from this string. In place, the value of *__str* is inserted. If *__pos* is beyond end of string, out_of_range is thrown. If the length of the result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

reserve() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::reserve ()  [constexpr]
```

Equivalent to shrink_to_fit().
References capacity(), get_allocator(), and length().
Referenced by std::basic_string< _CharT >::push_back(), and std::basic_string< _CharT >::shrink_to_fit().

reserve() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::reserve ()
```

Equivalent to shrink_to_fit().

reserve() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::reserve (
            size_type __res_arg)  [constexpr]
```

Attempt to preallocate enough memory for specified number of characters.

Parameters

__res_arg	Number of characters required.

Exceptions

std::length_error	If *__res_arg* exceeds `max_size()`.

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than max_size(), length_error is thrown.
The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.
References capacity(), get_allocator(), and std::size().
Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >:
std::tr2::operator>>(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow().

reserve() [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::reserve (
            size_type __res_arg)
```

Attempt to preallocate enough memory for specified number of characters.

Parameters

__res_arg	Number of characters required.

Exceptions

std::length_error	If *__res_arg* exceeds `max_size()`.

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than max_size(), length_error is thrown.
The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

resize() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::resize (
            size_type __n)  [inline], [constexpr]
```

Resizes the string to the specified number of characters.

Parameters

__n	Number of characters the string should contain.

This function will resize the string to the specified length. If the new size is smaller than the string's current size the string is truncated, otherwise the string is extended and new characters are default-constructed. For basic types such as char, this means setting them to 0.

resize() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::resize (
            size_type __n)  [inline]
```

Resizes the string to the specified number of characters.

Parameters

_↩ _n	Number of characters the string should contain.

This function will resize the string to the specified length. If the new size is smaller than the string's current size the string is truncated, otherwise the string is extended and new characters are default-constructed. For basic types such as char, this means setting them to 0.

resize() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::resize (
            size_type __n,
            _CharT __c)  [constexpr]
```

Resizes the string to the specified number of characters.

Parameters

_↩ _n	Number of characters the string should contain.
_↩ _c	Character to fill any new elements.

This function will resize the string to the specified number of characters. If the number is smaller than the string's current size the string is truncated, otherwise the string is extended and new elements are set to __c.
References append(), erase(), and size().
Referenced by std::basic_string< _CharT >::resize().

resize() [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::resize (
            size_type __n,
            _CharT __c)
```

Resizes the string to the specified number of characters.

Parameters

_↩ _n	Number of characters the string should contain.
_↩ _c	Character to fill any new elements.

This function will resize the string to the specified number of characters. If the number is smaller than the string's current size the string is truncated, otherwise the string is extended and new elements are set to __c.

rfind() [1/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::rfind (
            _CharT __c,
            size_type __pos = npos) const  [nodiscard], [constexpr], [noexcept]
```

Find last position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for *__c* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and std::size().

rfind() [2/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::rfind (
            _CharT __c,
            size_type __pos = npos) const  [noexcept]
```

Find last position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for *__c* within this string. If found, returns the index where it was found. If not found, returns npos.

rfind() [3/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::rfind (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [nodiscard], [constexpr], [noexcept]
```

Find last position of a C substring.

Parameters

__s	C string to locate.

__pos	Index of character to search back from.
__n	Number of characters from s to search for.

Returns

Index of start of last occurrence.

Starting from *__pos*, searches backward for the first *__n* characters in *__s* within this string. If found, returns the index where it begins. If not found, returns npos.
References std::min(), npos, and std::size().

rfind() [4/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::rfind (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [noexcept]
```

Find last position of a C substring.

Parameters

__s	C string to locate.
__pos	Index of character to search back from.
__n	Number of characters from s to search for.

Returns

Index of start of last occurrence.

Starting from *__pos*, searches backward for the first *__n* characters in *__s* within this string. If found, returns the index where it begins. If not found, returns npos.

rfind() [5/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::rfind (
            const _CharT * __s,
            size_type __pos = npos) const  [inline], [nodiscard], [constexpr]
```

Find last position of a C string.

Parameters

__s	C string to locate.
__pos	Index of character to start search at (default end).

Returns

Index of start of last occurrence.

Starting from *__pos*, searches backward for the value of *__s* within this string. If found, returns the index where it begins. If not found, returns npos.

rfind() [6/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::rfind (
            const _CharT * __s,
            size_type __pos = npos) const  [inline], [noexcept]
```

Find last position of a C string.

Parameters

__s	C string to locate.
__pos	Index of character to start search at (default end).

Returns

Index of start of last occurrence.

Starting from *__pos*, searches backward for the value of *__s* within this string. If found, returns the index where it begins. If not found, returns npos.

rfind() [7/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::rfind (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos = npos) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a string.

Parameters

__str	String to locate.
__pos	Index of character to search back from (default end).

Returns

Index of start of last occurrence.

Starting from *__pos*, searches backward for value of *__str* within this string. If found, returns the index where it begins. If not found, returns npos.
Referenced by std::basic_string< _CharT >::find_last_of(), std::basic_string< _CharT >::rfind(), std::basic_string< _CharT >::rfind(), std::basic_string< _CharT >::rfind(), and std::basic_string< _CharT >::rfind().

rfind() [8/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::rfind (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos = npos) const  [inline], [noexcept]
```

Find last position of a string.

Parameters

__str	String to locate.
__pos	Index of character to search back from (default end).

Returns

Index of start of last occurrence.

Starting from *__pos*, searches backward for value of *__str* within this string. If found, returns the index where it begins. If not found, returns npos.

shrink_to_fit() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::shrink_to_fit ()  [inline], [constexpr], [noexcept]
```

A non-binding request to reduce capacity() to size().

shrink_to_fit() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::shrink_to_fit ()  [inline], [noexcept]
```

A non-binding request to reduce capacity() to size().

size() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::size () const  [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns the number of characters in the string, not including any null-termination.

Referenced by std::basic_string< _CharT >::append(), std::basic_string< _CharT >::append(), std::basic_regex< char >::assign(), std::basic_string< _CharT >::assign(), std::basic_string< _CharT >::assign(), std::basic_string< _CharT >::assign(), std::basic_string< _CharT >::assign(), std::basic_string< _CharT >::assign(), std::basic_string< _CharT >::at(), std::basic_string< _CharT >::at(), std::basic_string< _CharT >::back(), std::basic_string< _CharT > std::basic_string< _CharT >::back(), std::basic_string< _CharT >::cend(), std::basic_string< _CharT >::compare(), std::basic_string< _CharT >::compare(), std::basic_string< _CharT >::compare(), std::basic_string< _CharT >::empty(), std::basic_string< _CharT >::end(), std::basic_string< _CharT >::end(), std::basic_string< _CharT >::end(), std::basic_string< _CharT >::find(), std::basic_string< _CharT >::find_first_not_of(), std::basic_string< _CharT >::find_first_of(), __gnu_debug::basic_string< char >::find_last_of(), std::basic_string< _CharT >::find_last_of(), std::match_results< _BidirectionalIterat std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::from_bytes(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::length(), std::operator+(), std::operator+(), std::operator+(), std::operator+(), std::operator+(), std::operator==(), std::tr2::operator>>(), std::basic_string< _CharT >::operator[](), std::basic_string< _CharT >::operator[](), std::basic_string< _CharT >::pop_back(), std::basic_string< _CharT >::pop_back(), std::basic_string< _CharT >::push_back(), std::basic_string< _CharT >::replace(), std::basic_string< _CharT >::replace(), resize(), std::wstring_convert< _Codecvt, _Elem, _Wide std::regex_traits< _CharT >::transform(), and std::regex_traits< _CharT >::transform_primary().

size() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::size () const  [inline], [noexcept]
```

Returns the number of characters in the string, not including any null-termination.

substr() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string std::basic_string< _CharT, _Traits, _Alloc >::substr (
            size_type __pos = 0,
            size_type __n = npos) const  [inline], [nodiscard], [constexpr]
```

Get a substring.

Parameters

__pos	Index of first character (default 0).
__n	Number of characters in substring (default remainder).

Returns

The new string.

Exceptions

std::out_of_range	If __pos > size().

Construct and return a new string using the *__n* characters starting at *__pos*. If the string is too short, use the remainder of the characters. If *__pos* is beyond the end of the string, out_of_range is thrown.

substr() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string std::basic_string< _CharT, _Traits, _Alloc >::substr (
            size_type __pos = 0,
            size_type __n = npos) const  [inline]
```

Get a substring.

Parameters

__pos	Index of first character (default 0).
__n	Number of characters in substring (default remainder).

Returns

The new string.

Exceptions

std::out_of_range	If __pos > size().

Construct and return a new string using the *__n* characters starting at *__pos*. If the string is too short, use the remainder of the characters. If *__pos* is beyond the end of the string, out_of_range is thrown.

swap() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::swap (
            basic_string< _CharT, _Traits, _Alloc > & __s)  [constexpr], [noexcept]
```

Swap contents with another string.

Parameters

__s	String to swap with.

Exchanges the contents of this string with that of *__s* in constant time.
References basic_string(), and get_allocator().
Referenced by std::basic_string< _CharT >::assign(), and std::basic_string< _CharT >::operator=().

swap() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::swap (
            basic_string< _CharT, _Traits, _Alloc > & __s)  [noexcept]
```

Swap contents with another string.

Parameters

_↩ _s	String to swap with.

Exchanges the contents of this string with that of __s in constant time.

5.233.4 Member Data Documentation

npos

```
template<typename _CharT, typename _Traits, typename _Alloc>
static const size_type std::basic_string< _CharT, _Traits, _Alloc >::npos  [static]
```

Value returned by various member functions when they fail.
Referenced by find(), find(), find_first_not_of(), find_first_not_of(), find_first_of(), find_last_not_of(), find_last_not_of(), find_last_of(), rfind(), and rfind().
The documentation for this class was generated from the following files:

- basic_string.h
- cow_string.h
- basic_string.tcc

5.234 std::basic_string_view< _CharT, _Traits > Class Template Reference

```
#include <>>
```

Public Types

- using **const_iterator**
- using **const_pointer**
- using **const_reference**
- using **const_reverse_iterator**
- using **difference_type**
- using **iterator**
- using **pointer**
- using **reference**
- using **reverse_iterator**
- using **size_type**
- using **traits_type**
- using **value_type**

Public Member Functions

- constexpr **basic_string_view** (const _CharT *__str) noexcept
- constexpr **basic_string_view** (const _CharT *__str, size_type __len) noexcept
- constexpr **basic_string_view** (const basic_string_view &) noexcept=default
- constexpr const_reference **at** (size_type __pos) const

- constexpr const_reference **back** () const noexcept
- constexpr const_iterator **begin** () const noexcept
- constexpr const_iterator **cbegin** () const noexcept
- constexpr const_iterator **cend** () const noexcept
- constexpr int **compare** (basic_string_view __str) const noexcept
- constexpr int **compare** (const _CharT *__str) const noexcept
- constexpr int **compare** (size_type __pos1, size_type __n1, basic_string_view __str) const
- constexpr int **compare** (size_type __pos1, size_type __n1, basic_string_view __str, size_type __pos2, size_type __n2) const
- constexpr int **compare** (size_type __pos1, size_type __n1, const _CharT *__str) const
- constexpr int **compare** (size_type __pos1, size_type __n1, const _CharT *__str, size_type __n2) const noexcept(false)
- constexpr size_type **copy** (_CharT *__str, size_type __n, size_type __pos=0) const
- constexpr const_reverse_iterator **crbegin** () const noexcept
- constexpr const_reverse_iterator **crend** () const noexcept
- constexpr const_pointer **data** () const noexcept
- constexpr bool **empty** () const noexcept
- constexpr const_iterator **end** () const noexcept
- constexpr size_type **find** (_CharT __c, size_type __pos=0) const noexcept
- constexpr size_type **find** (basic_string_view __str, size_type __pos=0) const noexcept
- constexpr size_type **find** (const _CharT *__str, size_type __pos, size_type __n) const noexcept
- constexpr size_type **find** (const _CharT *__str, size_type __pos=0) const noexcept
- constexpr size_type **find_first_not_of** (_CharT __c, size_type __pos=0) const noexcept
- constexpr size_type **find_first_not_of** (basic_string_view __str, size_type __pos=0) const noexcept
- constexpr size_type **find_first_not_of** (const _CharT *__str, size_type __pos, size_type __n) const noexcept
- constexpr size_type **find_first_not_of** (const _CharT *__str, size_type __pos=0) const noexcept
- constexpr size_type **find_first_of** (_CharT __c, size_type __pos=0) const noexcept
- constexpr size_type **find_first_of** (basic_string_view __str, size_type __pos=0) const noexcept
- constexpr size_type **find_first_of** (const _CharT *__str, size_type __pos, size_type __n) const noexcept
- constexpr size_type **find_first_of** (const _CharT *__str, size_type __pos=0) const noexcept
- constexpr size_type **find_last_not_of** (_CharT __c, size_type __pos=npos) const noexcept
- constexpr size_type **find_last_not_of** (basic_string_view __str, size_type __pos=npos) const noexcept
- constexpr size_type **find_last_not_of** (const _CharT *__str, size_type __pos, size_type __n) const noexcept
- constexpr size_type **find_last_not_of** (const _CharT *__str, size_type __pos=npos) const noexcept
- constexpr size_type **find_last_of** (_CharT __c, size_type __pos=npos) const noexcept
- constexpr size_type **find_last_of** (basic_string_view __str, size_type __pos=npos) const noexcept
- constexpr size_type **find_last_of** (const _CharT *__str, size_type __pos, size_type __n) const noexcept
- constexpr size_type **find_last_of** (const _CharT *__str, size_type __pos=npos) const noexcept
- constexpr const_reference **front** () const noexcept
- constexpr size_type **length** () const noexcept
- constexpr size_type **max_size** () const noexcept
- constexpr basic_string_view & **operator=** (const basic_string_view &) noexcept=default
- constexpr const_reference **operator[ ]** (size_type __pos) const noexcept
- constexpr const_reverse_iterator **rbegin** () const noexcept
- constexpr void **remove_prefix** (size_type __n) noexcept
- constexpr void **remove_suffix** (size_type __n) noexcept
- constexpr const_reverse_iterator **rend** () const noexcept
- constexpr size_type **rfind** (_CharT __c, size_type __pos=npos) const noexcept
- constexpr size_type **rfind** (basic_string_view __str, size_type __pos=npos) const noexcept
- constexpr size_type **rfind** (const _CharT *__str, size_type __pos, size_type __n) const noexcept
- constexpr size_type **rfind** (const _CharT *__str, size_type __pos=npos) const noexcept
- constexpr size_type **size** () const noexcept
- constexpr basic_string_view **substr** (size_type __pos=0, size_type __n=npos) const noexcept(false)
- constexpr void **swap** (basic_string_view &__sv) noexcept

Static Public Attributes

- static constexpr size_type **npos**

5.234.1 Detailed Description

template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
class std::basic_string_view< _CharT, _Traits >

A non-owning reference to a string.

Template Parameters

_CharT	Type of character
_Traits	Traits for character type, defaults to char_traits<_CharT>.

A basic_string_view looks like this:

```
_CharT*     _M_str
size_t      _M_len
```

The documentation for this class was generated from the following files:

- string_view
- bits/string_view.tcc

5.235 std::experimental::filesystem::v1::basic_string_view< _CharT, _Traits > Class Template Reference

`#include <>`

Public Types

- using **const_iterator**
- using **const_pointer**
- using **const_reference**
- using **const_reverse_iterator**
- using **difference_type**
- using **iterator**
- using **pointer**
- using **reference**
- using **reverse_iterator**
- using **size_type**
- using **traits_type**
- using **value_type**

Public Member Functions

- constexpr **basic_string_view** (const _CharT *__str) noexcept
- constexpr **basic_string_view** (const _CharT *__str, size_type __len) noexcept
- constexpr **basic_string_view** (const basic_string_view &) noexcept=default
- constexpr const_reference **at** (size_type __pos) const
- constexpr const_reference **back** () const noexcept
- constexpr const_iterator **begin** () const noexcept
- constexpr const_iterator **cbegin** () const noexcept
- constexpr const_iterator **cend** () const noexcept

- constexpr int **compare** (basic_string_view __str) const noexcept
- constexpr int **compare** (const _CharT *__str) const noexcept
- constexpr int **compare** (size_type __pos1, size_type __n1, basic_string_view __str) const
- constexpr int **compare** (size_type __pos1, size_type __n1, basic_string_view __str, size_type __pos2, size_type __n2) const
- constexpr int **compare** (size_type __pos1, size_type __n1, const _CharT *__str) const
- constexpr int **compare** (size_type __pos1, size_type __n1, const _CharT *__str, size_type __n2) const noexcept(false)
- constexpr size_type **copy** (_CharT *__str, size_type __n, size_type __pos=0) const
- constexpr const_reverse_iterator **crbegin** () const noexcept
- constexpr const_reverse_iterator **crend** () const noexcept
- constexpr const_pointer **data** () const noexcept
- constexpr bool **empty** () const noexcept
- constexpr const_iterator **end** () const noexcept
- constexpr size_type **find** (_CharT __c, size_type __pos=0) const noexcept
- constexpr size_type **find** (basic_string_view __str, size_type __pos=0) const noexcept
- constexpr size_type **find** (const _CharT *__str, size_type __pos, size_type __n) const noexcept
- constexpr size_type **find** (const _CharT *__str, size_type __pos=0) const noexcept
- constexpr size_type **find_first_not_of** (_CharT __c, size_type __pos=0) const noexcept
- constexpr size_type **find_first_not_of** (basic_string_view __str, size_type __pos=0) const noexcept
- constexpr size_type **find_first_not_of** (const _CharT *__str, size_type __pos, size_type __n) const noexcept
- constexpr size_type **find_first_not_of** (const _CharT *__str, size_type __pos=0) const noexcept
- constexpr size_type **find_first_of** (_CharT __c, size_type __pos=0) const noexcept
- constexpr size_type **find_first_of** (basic_string_view __str, size_type __pos=0) const noexcept
- constexpr size_type **find_first_of** (const _CharT *__str, size_type __pos, size_type __n) const noexcept
- constexpr size_type **find_first_of** (const _CharT *__str, size_type __pos=0) const noexcept
- constexpr size_type **find_last_not_of** (_CharT __c, size_type __pos=npos) const noexcept
- constexpr size_type **find_last_not_of** (basic_string_view __str, size_type __pos=npos) const noexcept
- constexpr size_type **find_last_not_of** (const _CharT *__str, size_type __pos, size_type __n) const noexcept
- constexpr size_type **find_last_not_of** (const _CharT *__str, size_type __pos=npos) const noexcept
- constexpr size_type **find_last_of** (_CharT __c, size_type __pos=npos) const noexcept
- constexpr size_type **find_last_of** (basic_string_view __str, size_type __pos=npos) const noexcept
- constexpr size_type **find_last_of** (const _CharT *__str, size_type __pos, size_type __n) const noexcept
- constexpr size_type **find_last_of** (const _CharT *__str, size_type __pos=npos) const noexcept
- constexpr const_reference **front** () const noexcept
- constexpr size_type **length** () const noexcept
- constexpr size_type **max_size** () const noexcept
- constexpr basic_string_view & **operator=** (const basic_string_view &) noexcept=default
- constexpr const_reference **operator[ ]** (size_type __pos) const noexcept
- constexpr const_reverse_iterator **rbegin** () const noexcept
- constexpr void **remove_prefix** (size_type __n) noexcept
- constexpr void **remove_suffix** (size_type __n) noexcept
- constexpr const_reverse_iterator **rend** () const noexcept
- constexpr size_type **rfind** (_CharT __c, size_type __pos=npos) const noexcept
- constexpr size_type **rfind** (basic_string_view __str, size_type __pos=npos) const noexcept
- constexpr size_type **rfind** (const _CharT *__str, size_type __pos, size_type __n) const noexcept
- constexpr size_type **rfind** (const _CharT *__str, size_type __pos=npos) const noexcept
- constexpr size_type **size** () const noexcept
- constexpr basic_string_view **substr** (size_type __pos=0, size_type __n=npos) const noexcept(false)
- constexpr void **swap** (basic_string_view &__sv) noexcept

Static Public Attributes

- static constexpr size_type **npos**

5.235.1 Detailed Description

template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
class std::experimental::filesystem::v1::basic_string_view< _CharT, _Traits >

A non-owning reference to a string.

Template Parameters

_CharT	Type of character
_Traits	Traits for character type, defaults to char_traits<_CharT>.

A basic_string_view looks like this:

```
_CharT*     _M_str
size_t      _M_len
```

The documentation for this class was generated from the following files:

- string_view
- bits/string_view.tcc

5.236 std::experimental::fundamentals_v1::basic_string_view< _CharT, _Traits > Class Template Reference

`#include <>>`

Public Types

- using **const_iterator**
- using **const_pointer**
- using **const_reference**
- using **const_reverse_iterator**
- using **difference_type**
- using **iterator**
- using **pointer**
- using **reference**
- using **reverse_iterator**
- using **size_type**
- using **traits_type**
- using **value_type**

Public Member Functions

- constexpr **basic_string_view** (const _CharT *__str)
- constexpr **basic_string_view** (const _CharT *__str, size_type __len)
- template<typename _Allocator>
  basic_string_view (const basic_string< _CharT, _Traits, _Allocator > &__str) noexcept
- constexpr **basic_string_view** (const basic_string_view &) noexcept=default
- constexpr const _CharT & **at** (size_type __pos) const
- constexpr const _CharT & **back** () const
- constexpr const_iterator **begin** () const noexcept

- constexpr const_iterator **cbegin** () const noexcept
- constexpr const_iterator **cend** () const noexcept
- constexpr int **compare** (basic_string_view __str) const noexcept
- constexpr int **compare** (const _CharT *__str) const noexcept
- constexpr int **compare** (size_type __pos1, size_type __n1, basic_string_view __str) const
- constexpr int **compare** (size_type __pos1, size_type __n1, basic_string_view __str, size_type __pos2, size_type __n2) const
- constexpr int **compare** (size_type __pos1, size_type __n1, const _CharT *__str) const
- constexpr int **compare** (size_type __pos1, size_type __n1, const _CharT *__str, size_type __n2) const
- size_type **copy** (_CharT *__str, size_type __n, size_type __pos=0) const
- const_reverse_iterator **crbegin** () const noexcept
- const_reverse_iterator **crend** () const noexcept
- constexpr const _CharT * **data** () const noexcept
- constexpr bool **empty** () const noexcept
- constexpr const_iterator **end** () const noexcept
- constexpr size_type **find** (_CharT __c, size_type __pos=0) const noexcept
- constexpr size_type **find** (basic_string_view __str, size_type __pos=0) const noexcept
- constexpr size_type **find** (const _CharT *__str, size_type __pos, size_type __n) const noexcept
- constexpr size_type **find** (const _CharT *__str, size_type __pos=0) const noexcept
- constexpr size_type **find_first_not_of** (_CharT __c, size_type __pos=0) const noexcept
- constexpr size_type **find_first_not_of** (basic_string_view __str, size_type __pos=0) const noexcept
- constexpr size_type **find_first_not_of** (const _CharT *__str, size_type __pos, size_type __n) const
- constexpr size_type **find_first_not_of** (const _CharT *__str, size_type __pos=0) const noexcept
- constexpr size_type **find_first_of** (_CharT __c, size_type __pos=0) const noexcept
- constexpr size_type **find_first_of** (basic_string_view __str, size_type __pos=0) const noexcept
- constexpr size_type **find_first_of** (const _CharT *__str, size_type __pos, size_type __n) const
- constexpr size_type **find_first_of** (const _CharT *__str, size_type __pos=0) const noexcept
- constexpr size_type **find_last_not_of** (_CharT __c, size_type __pos=npos) const noexcept
- constexpr size_type **find_last_not_of** (basic_string_view __str, size_type __pos=npos) const noexcept
- constexpr size_type **find_last_not_of** (const _CharT *__str, size_type __pos, size_type __n) const
- constexpr size_type **find_last_not_of** (const _CharT *__str, size_type __pos=npos) const noexcept
- constexpr size_type **find_last_of** (_CharT __c, size_type __pos=npos) const noexcept
- constexpr size_type **find_last_of** (basic_string_view __str, size_type __pos=npos) const noexcept
- constexpr size_type **find_last_of** (const _CharT *__str, size_type __pos, size_type __n) const
- constexpr size_type **find_last_of** (const _CharT *__str, size_type __pos=npos) const noexcept
- constexpr const _CharT & **front** () const
- constexpr size_type **length** () const noexcept
- constexpr size_type **max_size** () const noexcept
- template<typename _Allocator>
  operator basic_string< _CharT, _Traits, _Allocator > () const
- basic_string_view & **operator=** (const basic_string_view &) noexcept=default
- constexpr const _CharT & **operator[ ]** (size_type __pos) const
- const_reverse_iterator **rbegin** () const noexcept
- constexpr void **remove_prefix** (size_type __n)
- constexpr void **remove_suffix** (size_type __n)
- const_reverse_iterator **rend** () const noexcept
- constexpr size_type **rfind** (_CharT __c, size_type __pos=npos) const noexcept
- constexpr size_type **rfind** (basic_string_view __str, size_type __pos=npos) const noexcept
- constexpr size_type **rfind** (const _CharT *__str, size_type __pos, size_type __n) const noexcept
- constexpr size_type **rfind** (const _CharT *__str, size_type __pos=npos) const noexcept
- constexpr size_type **size** () const noexcept

- constexpr basic_string_view **substr** (size_type __pos=0, size_type __n=npos) const
- constexpr void **swap** (basic_string_view &__sv) noexcept
- template<typename _Allocator = std::allocator<_CharT>>
  basic_string< _CharT, _Traits, _Allocator > **to_string** (const _Allocator &__alloc=_Allocator()) const

Static Public Attributes

- static constexpr size_type **npos**

5.236.1 Detailed Description

template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
class std::experimental::fundamentals_v1::basic_string_view< _CharT, _Traits >

A non-owning reference to a string.

Template Parameters

_CharT	Type of character
_Traits	Traits for character type, defaults to char_traits<_CharT>.

A basic_string_view looks like this:

```
_CharT*     _M_str
size_t      _M_len
```

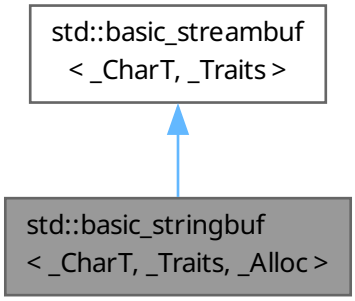
The documentation for this class was generated from the following files:

- experimental/string_view
- experimental/bits/string_view.tcc

5.237 std::basic_stringbuf< _CharT, _Traits, _Alloc > Class Template Reference

`#include <sstream>`

Inheritance diagram for std::basic_stringbuf< _CharT, _Traits, _Alloc >:



Public Types

- typedef __string_type::size_type **__size_type**

- typedef basic_streambuf< char_type, traits_type > **__streambuf_type**
- typedef basic_string< char_type, _Traits, _Alloc > **__string_type**
- typedef _Alloc **allocator_type**
- typedef _CharT **char_type**
- typedef traits_type::int_type **int_type**
- typedef traits_type::off_type **off_type**
- typedef traits_type::pos_type **pos_type**
- typedef _Traits **traits_type**

Public Member Functions

- basic_stringbuf ()
- **basic_stringbuf** (__string_type &&__s, ios_base::openmode __mode=ios_base::in|ios_base::out)
- **basic_stringbuf** (basic_stringbuf &&__rhs)
- **basic_stringbuf** (basic_stringbuf &&__rhs, const allocator_type &__a)
- basic_stringbuf (const __string_type &__str, ios_base::openmode __mode=ios_base::in|ios_base::out)
- **basic_stringbuf** (const allocator_type &__a)
- template<typename _SAlloc>
  basic_stringbuf (const basic_string< _CharT, _Traits, _SAlloc > &__s, const allocator_type &__a)
- template<typename _SAlloc>
  basic_stringbuf (const basic_string< _CharT, _Traits, _SAlloc > &__s, ios_base::openmode __mode, const allocator_type &__a)
- template<typename _SAlloc>
  basic_stringbuf (const basic_string< _CharT, _Traits, _SAlloc > &__s, ios_base::openmode __↩
  mode=ios_base::in|ios_base::out)
- **basic_stringbuf** (const basic_stringbuf &)=delete
- basic_stringbuf (ios_base::openmode __mode)
- **basic_stringbuf** (ios_base::openmode __mode, const allocator_type &__a)
- allocator_type **get_allocator** () const noexcept
- locale getloc () const
- streamsize in_avail ()
- basic_stringbuf & **operator=** (basic_stringbuf &&__rhs)
- basic_stringbuf & **operator=** (const basic_stringbuf &)=delete
- locale pubimbue (const locale &__loc)
- int_type sbumpc ()
- int_type sgetc ()
- streamsize sgetn (char_type *__s, streamsize __n)
- int_type snextc ()
- int_type sputbackc (char_type __c)
- int_type sputc (char_type __c)
- streamsize sputn (const char_type *__s, streamsize __n)
- __string_type **str** () &&
- __string_type str () const &
- void **str** (__string_type &&__s)
- void str (const __string_type &__s)
- template<__allocator_like _SAlloc>
  basic_string< _CharT, _Traits, _SAlloc > **str** (const _SAlloc &__sa) const
- template<__allocator_like _SAlloc>
  requires (!is_same_v<_SAlloc, _Alloc>)
  void **str** (const basic_string< _CharT, _Traits, _SAlloc > &__s)
- int_type sungetc ()
- void **swap** (basic_stringbuf &__rhs) noexcept(_Noexcept_swap::value)

- basic_string_view< char_type, traits_type > **view** () const noexcept

- basic_streambuf * pubsetbuf (char_type *__s, streamsize __n)
- pos_type pubseekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)
- pos_type pubseekpos (pos_type __sp, ios_base::openmode __mode=ios_base::in|ios_base::out)
- int pubsync ()

Protected Member Functions

- void **__safe_gbump** (streamsize __n)
- void **__safe_pbump** (streamsize __n)
- void **_M_pbump** (char_type *__pbeg, char_type *__pend, off_type __off)
- void **_M_stringbuf_init** (ios_base::openmode __mode)
- void **_M_sync** (char_type *__base, __size_type __i, __size_type __o)
- void **_M_update_egptr** ()
- void gbump (int __n)
- virtual void imbue (const locale &__loc)
- virtual int_type overflow (int_type __c=traits_type::eof())
- virtual int_type pbackfail (int_type __c=traits_type::eof())
- void pbump (int __n)
- virtual pos_type seekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)
- virtual pos_type seekpos (pos_type __sp, ios_base::openmode __mode=ios_base::in|ios_base::out)
- virtual __streambuf_type * setbuf (char_type *__s, streamsize __n)
- void setg (char_type *__gbeg, char_type *__gnext, char_type *__gend)
- void setp (char_type *__pbeg, char_type *__pend)
- virtual streamsize showmanyc ()
- void **swap** (basic_streambuf &__sb)
- virtual int sync ()
- virtual int_type uflow ()
- virtual int_type underflow ()
- virtual streamsize xsgetn (char_type *__s, streamsize __n)
- virtual streamsize xsputn (const char_type *__s, streamsize __n)

- char_type * eback () const
- char_type * gptr () const
- char_type * egptr () const

- char_type * pbase () const
- char_type * pptr () const
- char_type * epptr () const

Protected Attributes

- locale _M_buf_locale
- char_type * _M_in_beg
- char_type * _M_in_cur
- char_type * _M_in_end
- ios_base::openmode _M_mode
- char_type * _M_out_beg
- char_type * _M_out_cur
- char_type * _M_out_end
- __string_type **_M_string**

5.237.1 Detailed Description

template<typename _CharT, typename _Traits, typename _Alloc>
class std::basic_stringbuf< _CharT, _Traits, _Alloc >

The actual work of input and output (for std::string).

Template Parameters

_CharT	Type of character stream.
_Traits	Traits for character type, defaults to char_traits<_CharT>.
_Alloc	Allocator type, defaults to allocator<_CharT>.

This class associates either or both of its input and output sequences with a sequence of characters, which can be initialized from, or made available as, a `std::basic_string`. (Paraphrased from [27.7.1]/1.)
For this class, open modes (of type `ios_base::openmode`) have `in` set if the input sequence can be read, and `out` set if the output sequence can be written.

5.237.2 Constructor & Destructor Documentation

basic_stringbuf() [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_stringbuf< _CharT, _Traits, _Alloc >::basic_stringbuf ()  [inline]
```

Starts with an empty string buffer.
The default constructor initializes the parent class using its own default ctor.

basic_stringbuf() [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_stringbuf< _CharT, _Traits, _Alloc >::basic_stringbuf (
            ios_base::openmode __mode)  [inline], [explicit]
```

Starts with an empty string buffer.

Parameters

__mode	Whether the buffer can read, or write, or both.

The default constructor initializes the parent class using its own default ctor.

basic_stringbuf() [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_stringbuf< _CharT, _Traits, _Alloc >::basic_stringbuf (
            const __string_type & __str,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [inline], [explicit]
```

Starts with an existing string buffer.

Parameters

__str	A string to copy as a starting buffer.
__mode	Whether the buffer can read, or write, or both.

This constructor initializes the parent class using its own default ctor.

5.237.3 Member Function Documentation

eback()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::eback () const  [inline], [protected], [inherited]
```

Access to the get area.
These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_stringbuf< _Cha std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_filebuf< _CharT, _Traits >::underflow(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

egptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::egptr () const  [inline], [protected], [inherited]
```

Access to the get area.
These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Referenced by std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_filebuf< _CharT, _Traits >::underflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), std::wbuffer_convert< _Codecvt, _Elem, _Tr >::underflow(), std::basic_filebuf< _CharT, _Traits >::xsgetn(), and xsgetn().

epptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::epptr () const  [inline], [protected], [inherited]
```

Access to the put area.
These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_filebuf< _CharT, _Traits >::xsputn(), and xsputn().

gbump()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::gbump (
            int __n)  [inline], [protected], [inherited]
```

Moving the read position.

Parameters

_↩ _n	The delta by which to move.

This just advances the read position without returning any data.
Referenced by std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

getloc()

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::getloc () const  [inline], [inherited]
```

Locale access.

Returns

The current locale in effect.

If pubimbue(loc) has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

gptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::gptr () const  [inline], [protected], [inherited]
```

Access to the get area.
These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_stringbuf< _Cha std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_filebuf< _CharT, _Traits >::underflow(), std::basic_stringbuf< _CharT, _T std::wbuffer_convert< _Codecvt, _Elem, _Tr >::underflow(), std::basic_filebuf< _CharT, _Traits >::xsgetn(), and xsgetn().

imbue()

```
template<typename _CharT, typename _Traits>
virtual void std::basic_streambuf< _CharT, _Traits >::imbue (
            const locale & __loc)  [inline], [protected], [virtual], [inherited]
```

Changes translations.

Parameters

__loc	A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

Note

Base class version does nothing.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< _CharT, std::char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< wchar_t >, and std::basic_filebuf< wchar_t >.

in_avail()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::in_avail ()  [inline], [inherited]
```

Looking ahead into the stream.

Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

overflow()

```
template<class _CharT, class _Traits, class _Alloc>
basic_stringbuf< _CharT, _Traits, _Alloc >::int_type std::basic_stringbuf< _CharT, _Traits, _↩
Alloc >::overflow (
            int_type __c = traits_type::eof())  [protected], [virtual]
```

Consumes data from the buffer; writes to the controlled sequence.

Parameters

_↩ _c	An additional character to consume.

Returns

eof() to indicate failure, something else (usually __c, or not_eof())

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if __c is not `eof()`.
For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.
A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_mode, std::basic_string< _CharT, _Traits, _Alloc >::assign(), std::basic_streambuf< _CharT, _Traits >::eback(), std::basic_streambuf< _CharT, _Traits >::egptr(), std::basic_streambuf< _CharT, _Traits >::epptr(), std::basic_streambuf< _CharT, _Tra std::ios_base::in, std::max(), std::min(), std::ios_base::out, std::basic_streambuf< _CharT, _Traits >::pbase(), std::basic_streambuf< _CharT, _Traits >::pbump(), std::basic_streambuf< _CharT, _Traits >::pptr(), std::basic_string< _CharT, _Traits, std::basic_string< _CharT, _Traits, _Alloc >::reserve(), and std::basic_streambuf< _CharT, _Traits >::setg().

pbackfail()

```
template<class _CharT, class _Traits, class _Alloc>
basic_stringbuf< _CharT, _Traits, _Alloc >::int_type std::basic_stringbuf< _CharT, _Traits, _↩
Alloc >::pbackfail (
            int_type __c = traits_type::eof())  [protected], [virtual]
```

Tries to back up the input sequence.

Parameters

_↩ _c	The character to be inserted back into the sequence.

Returns

eof() on failure, *some other value* on success

Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_mode, std::basic_streambuf< _CharT, _Traits >::eback(), std::basic_streambuf< _CharT, _Traits >::gbump(), std::basic_streambuf< _CharT, _Traits >::gptr(), and std::ios_base::out.

pbase()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::pbase () const  [inline], [protected], [inherited]
```

Access to the put area.
These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, std::basic_filebuf< _CharT, _Traits >::sync(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

pbump()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::pbump (
            int __n)  [inline], [protected], [inherited]
```

Moving the write position.

Parameters

_↩ _n	The delta by which to move.

This just advances the write position without returning any data.
Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow().

pptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::pptr () const  [inline], [protected], [inherited]
```

Access to the put area.

These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_filebuf< _CharT, _T std::basic_filebuf< _CharT, _Traits >::xsputn(), and xsputn().

pubimbue()

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::pubimbue (
            const locale & __loc)  [inline], [inherited]
```

Entry point for imbue().

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls the derived imbue(__loc).

pubseekoff()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekoff (
            off_type __off,
            ios_base::seekdir __way,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [inline], [inherited]
```

Alters the stream position.

Parameters

__off	Offset.
__way	Value for ios_base::seekdir.
__mode	Value for ios_base::openmode.

Calls virtual seekoff function.

pubseekpos()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekpos (
            pos_type __sp,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [inline], [inherited]
```

Alters the stream position.

Parameters

__sp	Position
__mode	Value for ios_base::openmode.

Calls virtual seekpos function.

pubsetbuf()

```
template<typename _CharT, typename _Traits>
basic_streambuf * std::basic_streambuf< _CharT, _Traits >::pubsetbuf (
            char_type * __s,
            streamsize __n)  [inline], [inherited]
```

Entry points for derived buffer functions.
The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

pubsync()

```
template<typename _CharT, typename _Traits>
int std::basic_streambuf< _CharT, _Traits >::pubsync ()  [inline], [inherited]
```

Calls virtual sync function.
Referenced by std::basic_istream< _CharT, _Traits >::sync().

sbumpc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sbumpc ()  [inline], [inherited]
```

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.
Referenced by std::basic_istream< _CharT, _Traits >::getline().

seekoff()

```
template<class _CharT, class _Traits, class _Alloc>
basic_stringbuf< _CharT, _Traits, _Alloc >::pos_type std::basic_stringbuf< _CharT, _Traits, _↩
Alloc >::seekoff (
            off_type ,
            ios_base::seekdir ,
            ios_base::openmode  = ios_base::in | ios_base::out)  [protected], [virtual]
```

Alters the stream positions.
Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_mode, std::ios_base::cur, std::basic_streambuf< _CharT, _Traits >::eback(), std::basic_streambuf< _CharT, _Traits >::e
std::ios_base::end, std::basic_streambuf< _CharT, _Traits >::epptr(), std::basic_streambuf< _CharT, _Traits >::gptr(),
std::ios_base::in, std::ios_base::out, std::basic_streambuf< _CharT, _Traits >::pbase(), std::basic_streambuf< _CharT, _Traits >::pptr(),
and std::basic_streambuf< _CharT, _Traits >::setg().

seekpos()

```
template<class _CharT, class _Traits, class _Alloc>
basic_stringbuf< _CharT, _Traits, _Alloc >::pos_type std::basic_stringbuf< _CharT, _Traits, _↩
Alloc >::seekpos (
            pos_type ,
            ios_base::openmode  = ios_base::in | ios_base::out)  [protected], [virtual]
```

Alters the stream positions.
Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_mode, std::basic_streambuf< _CharT, _Traits >::eback(), std::basic_streambuf< _CharT, _Traits >::egptr(), std::basic_streambuf< _CharT, _Traits >::epptr(), std::ios_base::in, std::ios_base::out, std::basic_streambuf< _CharT, _Traits >::pbase() and std::basic_streambuf< _CharT, _Traits >::setg().

setbuf()

```
template<typename _CharT, typename _Traits, typename _Alloc>
virtual __streambuf_type * std::basic_stringbuf< _CharT, _Traits, _Alloc >::setbuf (
            char_type * __s,
            streamsize __n)  [inline], [protected], [virtual]
```

Manipulates the buffer.

Parameters

_↩ _s	Pointer to a buffer area.
_↩ _n	Size of __s.

Returns

`this`

If no buffer has already been created, and both __s and __n are non-zero, then __s is used as a buffer; see https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html#io.streambuf.↩ buffering for more.
Reimplemented from std::basic_streambuf< _CharT, _Traits >.

setg()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setg (
            char_type * __gbeg,
            char_type * __gnext,
            char_type * __gend)  [inline], [protected], [inherited]
```

Setting the three read area pointers.

Parameters

__gbeg	A pointer.
__gnext	A pointer.

__gend	A pointer.

Postcondition

__gbeg == `eback()`, *__gnext* == `gptr()`, and *__gend* == `egptr()`

Referenced by std::wbuffer_convert< _Codecvt, _Elem, _Tr >::wbuffer_convert(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflo
std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(),
and std::basic_filebuf< _CharT, _Traits >::xsgetn().

setp()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setp (
            char_type * __pbeg,
            char_type * __pend)  [inline], [protected], [inherited]
```

Setting the three write area pointers.

Parameters

__pbeg	A pointer.
__pend	A pointer.

Postcondition

__pbeg == `pbase()`, *__pbeg* == `pptr()`, and *__pend* == `epptr()`

Referenced by std::wbuffer_convert< _Codecvt, _Elem, _Tr >::wbuffer_convert().

sgetc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sgetc ()  [inline], [inherited]
```

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.
Referenced by std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(),
std::basic_istream< _CharT, _Traits >::ignore(), and std::basic_istream< _CharT, _Traits >::ignore().

sgetn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sgetn (
            char_type * __s,
            streamsize __n)  [inline], [inherited]
```

Entry point for xsgetn.

Parameters

_↩ _s	A buffer area.
_↩ _n	A count.

Returns xsgetn(__s,__n). The effect is to fill __s[0] through __s[__n-1] with characters from the input sequence, if possible.

showmanyc()

```
template<typename _CharT, typename _Traits, typename _Alloc>
virtual streamsize std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc ()  [inline], [protected],
[virtual]
```

Investigating the data available.

Returns

An estimate of the number of characters available in the input sequence, or -1.

If it returns a positive value, then successive calls to `underflow()` *will not return* `traits::eof()` *until at least that number of characters have been supplied. If* `showmanyc()` *returns -1, then calls to* `underflow()` *or* `uflow()` *will fail.* [27.5.2.4.3]/1

Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to underflow or uflow] will not return* `eof()` *but that they will return immediately.*

The standard adds that *the morphemes of* `showmanyc` *are* ***es-how-many-see****, not* ***show-manic****.*

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

snextc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::snextc ()  [inline], [inherited]
```

Getting the next character.

Returns

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.
Referenced by std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< char_type, traits_type >::ignore(), std::basic_istream< char_type, traits_type >::operator>>(), and std::basic_istream< char_type, traits_type >::putback().

sputbackc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputbackc (
            char_type __c)  [inline], [inherited]
```

Pushing characters back into the input stream.

Parameters

_↩ _c	The character to push back.

Returns

The previous character, if possible.

Similar to sungetc(), but __c is pushed onto the stream instead of *the previous character.* If successful, the next character fetched from the input stream will be __c.
Referenced by std::basic_istream< _CharT, _Traits >::putback().

sputc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputc (
             char_type __c)  [inline], [inherited]
```

Entry point for all single-character output functions.

Parameters

_↩ _c	A character to output.

Returns

__c, if possible.

One of two public output functions.
If a write position is available for the output sequence (i.e., the buffer is not full), stores __c in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(↩ __c)`.
Referenced by std::wbuffer_convert< _Codecvt, _Elem, _Tr >::overflow().

sputn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sputn (
             const char_type * __s,
             streamsize __n)  [inline], [inherited]
```

Entry point for all single-character output functions.

Parameters

_↩ _s	A buffer read area.
_↩ _n	A count.

One of two public output functions.
Returns xsputn(__s,__n). The effect is to write __s[0] through __s[__n-1] to the output sequence, if possible.

str() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
__string_type std::basic_stringbuf< _CharT, _Traits, _Alloc >::str () const &  [inline], [nodiscard]
```

Copying out the string buffer.

Returns

A copy of one of the underlying sequences.

If the buffer is only created in input mode, the underlying character sequence is equal to the input sequence; otherwise, it is equal to the output sequence. [27.7.1.2]/1

str() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_stringbuf< _CharT, _Traits, _Alloc >::str (
            const __string_type & __s)  [inline]
```

Setting a new buffer.

Parameters

_↩ _s	The string to use as a new sequence.

Deallocates any previous stored sequence, then copies *s* to use as a new one.

sungetc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sungetc ()  [inline], [inherited]
```

Moving backwards in the input stream.

Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns pbackfail(). The effect is to *unget* the last character *gotten*.
Referenced by std::basic_istream< char_type, traits_type >::sentry::sentry().

sync()

```
template<typename _CharT, typename _Traits>
virtual int std::basic_streambuf< _CharT, _Traits >::sync ()  [inline], [protected], [virtual],
[inherited]
```

Synchronizes the buffer arrays with the controlled sequences.

Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

Note

Base class version does nothing, returns zero.

Reimplemented in __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, std::basic_filebuf< _CharT, std::char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< wchar_t >, std::basic_filebuf< wchar_t >, and std::wbuffer_convert< _Codecvt, _Elem, _Tr >.

uflow()

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::uflow ()  [inline], [protected], [virtual],
[inherited]
```

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.
Reimplemented in __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >.
Referenced by xsgetn().

underflow()

```
template<class _CharT, class _Traits, class _Alloc>
basic_stringbuf< _CharT, _Traits, _Alloc >::int_type std::basic_stringbuf< _CharT, _Traits, _↩
Alloc >::underflow ()  [protected], [virtual]
```

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.
For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.
A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_mode, std::basic_streambuf< _CharT, _Traits >::egptr(), std::basic_streambuf< _CharT, _Traits >::gptr(), and std::ios_base::in.

xsgetn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::xsgetn (
            char_type * __s,
            streamsize __n)  [protected], [virtual], [inherited]
```

Multiple character extraction.

Parameters

_↩ _s	A buffer area.
_↩ _n	Maximum number of characters to assign.

Returns

The number of characters assigned.

Fills __*s*[0] through __*s*[__n-1] with characters from the input sequence, as if by `sbumpc()`. Stops when either __*n* characters have been copied, or when `traits::eof()` would be copied.
It is expected that derived classes provide a more efficient implementation by overriding this definition.
Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< _CharT, std::char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< char_type, traits_type >, and std::basic_filebuf< wchar_t >.
References egptr(), gptr(), std::min(), and uflow().

xsputn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::xsputn (
            const char_type * __s,
            streamsize __n)  [protected], [virtual], [inherited]
```

Multiple character insertion.

Parameters

_↩ _*s*	A buffer area.
_↩ _*n*	Maximum number of characters to write.

Returns

The number of characters written.

Writes __*s*[0] through __*s*[__n-1] to the output sequence, as if by `sputc()`. Stops when either *n* characters have been copied, or when `sputc()` would return `traits::eof()`.
It is expected that derived classes provide a more efficient implementation by overriding this definition.
Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< _CharT, std::char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< char_type, traits_type >, and std::basic_filebuf< wchar_t >.
References epptr(), std::min(), overflow(), and pptr().

5.237.4 Member Data Documentation

_M_buf_locale

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::_M_buf_locale  [protected], [inherited]
```

Current locale setting.
Referenced by std::basic_filebuf< _CharT, _Traits >::basic_filebuf().

_M_in_beg

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_beg  [protected], [inherited]
```

Start of get area.

_M_in_cur

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_cur  [protected], [inherited]
```
Current read area.

_M_in_end

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_end  [protected], [inherited]
```
End of get area.

_M_mode

```
template<typename _CharT, typename _Traits, typename _Alloc>
ios_base::openmode std::basic_stringbuf< _CharT, _Traits, _Alloc >::_M_mode  [protected]
```
Place to stash in || out || in | out settings for current stringbuf.
Referenced by overflow(), pbackfail(), seekoff(), seekpos(), and underflow().

_M_out_beg

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_beg  [protected], [inherited]
```
Start of put area.

_M_out_cur

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_cur  [protected], [inherited]
```
Current put area.

_M_out_end

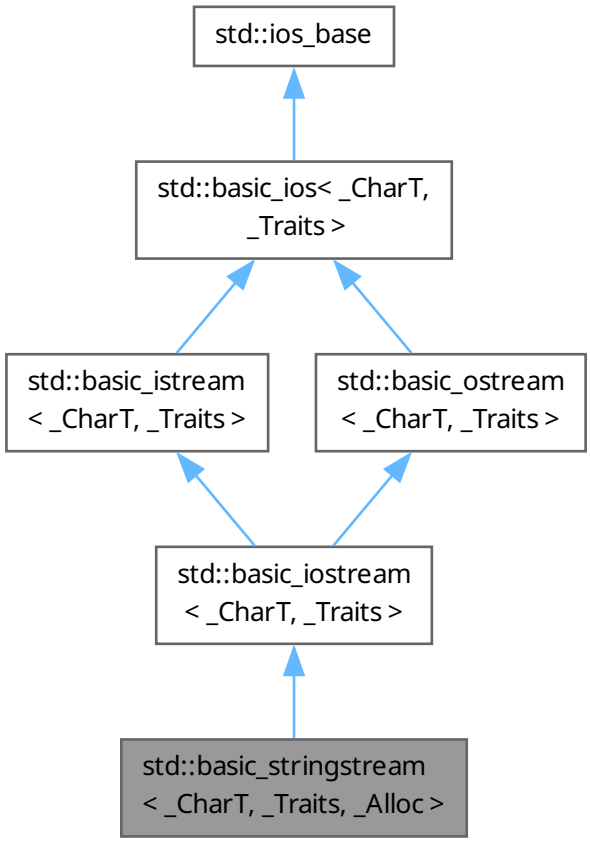
```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_end  [protected], [inherited]
```
End of put area.
The documentation for this class was generated from the following files:

- iosfwd
- sstream
- sstream.tcc

5.238 std::basic_stringstream< _CharT, _Traits, _Alloc > Class Template Reference

```
#include <sstream>
```

Inheritance diagram for std::basic_stringstream< _CharT, _Traits, _Alloc >:



Public Types

- typedef ctype< _CharT > **__ctype_type**
- typedef ctype< _CharT > **__ctype_type**
- typedef basic_ios< _CharT, _Traits > **__ios_type**
- typedef basic_ios< _CharT, _Traits > **__ios_type**
- typedef basic_iostream< char_type, traits_type > **__iostream_type**
- typedef basic_istream< _CharT, _Traits > **__istream_type**
- typedef num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > **__num_get_type**
- typedef num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > **__num_put_type**
- typedef basic_ostream< _CharT, _Traits > **__ostream_type**
- typedef basic_streambuf< _CharT, _Traits > **__streambuf_type**
- typedef basic_streambuf< _CharT, _Traits > **__streambuf_type**
- typedef basic_string< _CharT, _Traits, _Alloc > **__string_type**
- typedef basic_stringbuf< _CharT, _Traits, _Alloc > **__stringbuf_type**
- typedef _Alloc **allocator_type**
- typedef _CharT **char_type**

- enum event { **erase_event** , **imbue_event** , **copyfmt_event** }
- typedef void(∗ event_callback) (event __e, ios_base &__b, int __i)
- typedef traits_type::int_type **int_type**
- typedef _Ios_Iostate iostate
- typedef traits_type::off_type **off_type**
- typedef _Ios_Openmode openmode
- typedef traits_type::pos_type **pos_type**
- typedef _Ios_Seekdir seekdir
- typedef _Traits **traits_type**

Public Member Functions

- basic_stringstream ()
- **basic_stringstream** (__string_type &&__str, ios_base::openmode __mode=ios_base::in|ios_base::out)
- **basic_stringstream** (basic_stringstream &&__rhs)
- basic_stringstream (const __string_type &__str, ios_base::openmode __m=ios_base::out|ios_base::in)
- template<typename _SAlloc>
  basic_stringstream (const basic_string< _CharT, _Traits, _SAlloc > &__str, const allocator_type &__a)
- template<typename _SAlloc>
  basic_stringstream (const basic_string< _CharT, _Traits, _SAlloc > &__str, ios_base::openmode __mode, const allocator_type &__a)
- template<typename _SAlloc>
  basic_stringstream (const basic_string< _CharT, _Traits, _SAlloc > &__str, ios_base::openmode __↩ mode=ios_base::in|ios_base::out)
- **basic_stringstream** (const basic_stringstream &)=delete
- basic_stringstream (ios_base::openmode __m)
- **basic_stringstream** (ios_base::openmode __mode, const allocator_type &__a)
- ∼basic_stringstream ()
- template<typename _ValueT>
  basic_istream< _CharT, _Traits > & **_M_extract** (_ValueT &__v)
- const locale & _M_getloc () const
- template<typename _ValueT>
  basic_ostream< _CharT, _Traits > & **_M_insert** (_ValueT __v)
- void **_M_setstate** (iostate __state)
- bool bad () const
- void clear (iostate __state=goodbit)
- basic_ios & copyfmt (const basic_ios &__rhs)
- bool eof () const
- iostate exceptions () const
- void exceptions (iostate __except)
- bool fail () const
- char_type fill () const
- char_type fill (char_type __ch)
- fmtflags flags () const
- fmtflags flags (fmtflags __fmtfl)
- __ostream_type & flush ()
- streamsize gcount () const
- basic_istream< char > & getline (char_type ∗__s, streamsize __n, char_type __delim)
- basic_istream< wchar_t > & **getline** (char_type ∗__s, streamsize __n, char_type __delim)
- locale getloc () const
- bool good () const

- basic_istream< char > & **ignore** (streamsize __n)
- basic_istream< wchar_t > & **ignore** (streamsize __n)
- basic_istream< char > & **ignore** (streamsize __n, int_type __delim)
- basic_istream< wchar_t > & **ignore** (streamsize __n, int_type __delim)
- locale imbue (const locale &__loc)
- long & iword (int __ix)
- char narrow (char_type __c, char __dfault) const
- __ostream_type & operator<< (__streambuf_type *__sb)
- __ostream_type & operator<< (const void *__p)
- __ostream_type & **operator**<< (nullptr_t)
- basic_stringstream & **operator=** (basic_stringstream &&__rhs)
- basic_stringstream & **operator=** (const basic_stringstream &)=delete
- __istream_type & operator>> (__streambuf_type *__sb)
- __istream_type & operator>> (void *&__p)
- streamsize precision () const
- streamsize precision (streamsize __prec)
- void *& pword (int __ix)
- basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > *__sb)
- __stringbuf_type * rdbuf () const
- iostate rdstate () const
- void register_callback (event_callback __fn, int __index)
- __ostream_type & seekp (off_type, ios_base::seekdir)
- __ostream_type & seekp (pos_type)
- fmtflags setf (fmtflags __fmtfl)
- fmtflags setf (fmtflags __fmtfl, fmtflags __mask)
- void setstate (iostate __state)
- __string_type **str** () &&
- __string_type str () const &
- void **str** (__string_type &&__s)
- void str (const __string_type &__s)
- template<__allocator_like _SAlloc>
  basic_string< _CharT, _Traits, _SAlloc > **str** (const _SAlloc &__sa) const
- template<__allocator_like _SAlloc>
  requires (!is_same_v<_SAlloc, _Alloc>)
  void **str** (const basic_string< _CharT, _Traits, _SAlloc > &__s)
- void **swap** (basic_stringstream &__rhs)
- pos_type tellp ()
- basic_ostream< _CharT, _Traits > * tie () const
- basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > *__tiestr)
- void unsetf (fmtflags __mask)
- basic_string_view< char_type, traits_type > **view** () const noexcept
- char_type widen (char __c) const
- streamsize width () const
- streamsize width (streamsize __wide)

- __istream_type & operator>> (__istream_type &(*__pf)(__istream_type &))
- __istream_type & operator>> (__ios_type &(*__pf)(__ios_type &))
- __istream_type & operator>> (ios_base &(*__pf)(ios_base &))

Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type std::basic_istream::sentry with the second argument (noskipws) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, ios_base::badbit will be turned on in the stream's error state (without causing an ios_base::failure to be thrown) and the original exception will be rethrown if badbit is set in the exceptions mask.

- __istream_type & operator>> (bool &__n)
- __istream_type & operator>> (short &__n)
- __istream_type & operator>> (unsigned short &__n)
- __istream_type & operator>> (int &__n)
- __istream_type & operator>> (unsigned int &__n)
- __istream_type & operator>> (long &__n)
- __istream_type & operator>> (unsigned long &__n)
- __istream_type & operator>> (long long &__n)
- __istream_type & operator>> (unsigned long long &__n)

- __istream_type & operator>> (float &__f)
- __istream_type & operator>> (double &__f)
- __istream_type & operator>> (long double &__f)

Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type std::basic_istream::sentry with the second argument (noskipws) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by gcount().

If an exception is thrown during extraction, ios_base::badbit will be turned on in the stream's error state (without causing an ios_base::failure to be thrown) and the original exception will be rethrown if badbit is set in the exceptions mask.

- int_type get ()
- __istream_type & get (char_type &__c)
- __istream_type & get (char_type *__s, streamsize __n, char_type __delim)
- __istream_type & get (char_type *__s, streamsize __n)
- __istream_type & get (__streambuf_type &__sb, char_type __delim)
- __istream_type & get (__streambuf_type &__sb)
- __istream_type & getline (char_type *__s, streamsize __n, char_type __delim)
- __istream_type & getline (char_type *__s, streamsize __n)
- __istream_type & ignore (streamsize __n, int_type __delim)
- __istream_type & ignore (streamsize __n)
- __istream_type & ignore ()
- int_type peek ()
- __istream_type & read (char_type *__s, streamsize __n)
- streamsize readsome (char_type *__s, streamsize __n)
- __istream_type & putback (char_type __c)
- __istream_type & unget ()
- int sync ()
- pos_type tellg ()
- __istream_type & seekg (pos_type)
- __istream_type & seekg (off_type, ios_base::seekdir)

- operator bool () const
- bool operator! () const

- __ostream_type & operator<< (__ostream_type &(*__pf)(__ostream_type &))
- __ostream_type & operator<< (__ios_type &(*__pf)(__ios_type &))
- __ostream_type & operator<< (ios_base &(*__pf)(ios_base &))

Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type std::basic_ostream::sentry. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, ios_base::badbit will be turned on in the stream's error state without causing an ios_base::failure to be thrown. The original exception will then be rethrown.

- __ostream_type & operator<< (long __n)
- __ostream_type & operator<< (unsigned long __n)
- __ostream_type & operator<< (bool __n)
- __ostream_type & operator<< (short __n)
- __ostream_type & operator<< (unsigned short __n)
- __ostream_type & operator<< (int __n)
- __ostream_type & operator<< (unsigned int __n)
- __ostream_type & operator<< (long long __n)
- __ostream_type & operator<< (unsigned long long __n)

- __ostream_type & operator<< (double __f)
- __ostream_type & operator<< (float __f)
- __ostream_type & operator<< (long double __f)

Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type std::basic_ostream::sentry. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, ios_base::badbit will be turned on in the stream's error state. If badbit is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- __ostream_type & put (char_type __c)
- __ostream_type & write (const char_type *__s, streamsize __n)

Static Public Member Functions

- static bool sync_with_stdio (bool __sync=true)
- static int xalloc () throw ()

Public Attributes

- class __attribute((__abi_tag__("cxx11"))) failure typedef _Ios_Fmtflags fmtflags

Static Public Attributes

- static const openmode **__noreplace**
- static const fmtflags adjustfield
- static const openmode app
- static const openmode ate
- static const iostate badbit
- static const fmtflags basefield
- static const seekdir beg
- static const openmode binary
- static const fmtflags boolalpha
- static const seekdir cur
- static const fmtflags dec
- static const seekdir end
- static const iostate eofbit
- static const iostate failbit
- static const fmtflags fixed
- static const fmtflags floatfield
- static const iostate goodbit
- static const fmtflags hex
- static const openmode in
- static const fmtflags internal
- static const fmtflags left
- static const fmtflags oct
- static const openmode out
- static const fmtflags right
- static const fmtflags scientific
- static const fmtflags showbase
- static const fmtflags showpoint
- static const fmtflags showpos
- static const fmtflags skipws
- static const openmode trunc
- static const fmtflags unitbuf
- static const fmtflags uppercase

Protected Types

- enum { **_S_local_word_size** }

Protected Member Functions

- void **_M_cache_locale** (const locale &__loc)
- void **_M_call_callbacks** (event __ev) throw ()
- void **_M_dispose_callbacks** (void) throw ()
- template<typename _ValueT>
  __istream_type & **_M_extract** (_ValueT &__v)
- _Words & **_M_grow_words** (int __index, bool __iword)
- void **_M_init** () throw ()
- template<typename _ValueT>
  __ostream_type & **_M_insert** (_ValueT __v)
- void **_M_move** (ios_base &) noexcept
- void **_M_swap** (ios_base &__rhs) noexcept

- void init (basic_streambuf< _CharT, _Traits > *__sb)
- void **move** (basic_ios &&__rhs)
- void **move** (basic_ios &__rhs)
- void **set_rdbuf** (basic_streambuf< _CharT, _Traits > *__sb)
- void **swap** (basic_ios &__rhs) noexcept
- void **swap** (basic_iostream &__rhs)
- void **swap** (basic_istream &__rhs)
- void **swap** (basic_ostream &__rhs)

Protected Attributes

- _Callback_list * **_M_callbacks**
- const __ctype_type * **_M_ctype**
- iostate **_M_exception**
- char_type **_M_fill**
- bool **_M_fill_init**
- fmtflags **_M_flags**
- streamsize _M_gcount
- locale **_M_ios_locale**
- _Words **_M_local_word** [_S_local_word_size]
- const __num_get_type * **_M_num_get**
- const __num_put_type * **_M_num_put**
- streamsize **_M_precision**
- basic_streambuf< _CharT, _Traits > * **_M_streambuf**
- iostate **_M_streambuf_state**
- basic_ostream< _CharT, _Traits > * **_M_tie**
- streamsize **_M_width**
- _Words * **_M_word**
- int **_M_word_size**
- _Words **_M_word_zero**

5.238.1 Detailed Description

template<typename _CharT, typename _Traits, typename _Alloc>
class std::basic_stringstream< _CharT, _Traits, _Alloc >

Controlling input and output for std::string.

Template Parameters

_CharT	Type of character stream.
_Traits	Traits for character type, defaults to char_traits<_CharT>.
_Alloc	Allocator type, defaults to allocator<_CharT>.

This class supports reading from and writing to objects of type std::basic_string, using the inherited functions from std::basic_iostream. To control the associated sequence, an instance of std::basic_stringbuf is used, which this page refers to as `sb`.

5.238.2 Member Typedef Documentation

event_callback

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i)  [inherited]
```

The type of an event callback function.

Parameters

_↩ _e	One of the members of the event enum.
_↩ _b	Reference to the ios_base object.
_↩ _i	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several ios_base and basic_ios functions, specifically imbue(), copyfmt(), and ~ios().

iostate

```
typedef _Ios_Iostate std::ios_base::iostate  [inherited]
```

This is a bitmask type.
_Ios_Iostate is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit
- eofbit
- failbit
- goodbit

openmode

```
typedef _Ios_Openmode std::ios_base::openmode  [inherited]
```

This is a bitmask type.
_Ios_Openmode is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type openmode are:

- app
- ate
- binary
- in
- out
- trunc

seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir  [inherited]
```

This is an enumerated type.
_Ios_Seekdir is implementation-defined. Defined values of type seekdir are:

- beg
- cur, equivalent to SEEK_CUR in the C standard library.
- end, equivalent to SEEK_END in the C standard library.

5.238.3 Member Enumeration Documentation

event

```
enum std::ios_base::event  [inherited]
```

The set of events that may be passed to an event callback.
erase_event is used during ~ios() and copyfmt(). imbue_event is used during imbue(). copyfmt_event is used during copyfmt().

5.238.4 Constructor & Destructor Documentation

basic_stringstream() [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_stringstream< _CharT, _Traits, _Alloc >::basic_stringstream ()  [inline]
```

Default constructor starts with an empty string buffer.
Initializes `sb` using the mode `in|out`, and passes `&sb` to the base class initializer. Does not allocate any buffer.
That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

basic_stringstream() [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_stringstream< _CharT, _Traits, _Alloc >::basic_stringstream (
            ios_base::openmode __m)  [inline], [explicit]
```

Starts with an empty string buffer.

Parameters

_↩ _*m*	Whether the buffer can read, or write, or both.

Initializes `sb` using the mode from `__m`, and passes `&sb` to the base class initializer. Does not allocate any buffer.
That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

basic_stringstream() [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_stringstream< _CharT, _Traits, _Alloc >::basic_stringstream (
            const __string_type & __str,
            ios_base::openmode __m = ios_base::out | ios_base::in)  [inline], [explicit]
```

Starts with an existing string buffer.

Parameters

__str	A string to copy as a starting buffer.
__m	Whether the buffer can read, or write, or both.

Initializes `sb` using *__str* and `__m`, and passes `&sb` to the base class initializer.
That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

~basic_stringstream()

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_stringstream< _CharT, _Traits, _Alloc >::~basic_stringstream ()  [inline]
```

The destructor does nothing.
The buffer is deallocated by the stringbuf object, not the formatting stream.

5.238.5 Member Function Documentation

_M_getloc()

```
const locale & std::ios_base::_M_getloc () const  [inline], [inherited]
```

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.
Referenced by std::money_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter
std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIte
std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::num_put< _CharT, _OutIter
std::time_put< _CharT, _OutIter >::do_put(), std::time_get< _CharT, _InIter >::get(), and std::time_put< _CharT, _OutIter >::put().

bad()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry().

clear()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::clear (
            iostate __state = goodbit)  [inherited]
```

[Re]sets the error state.

Parameters

__state	The new state flag(s) to set.

See std::ios_base::iostate for the possible bit values. Most users will not need to pass an argument.
References std::ios_base::badbit, exceptions(), rdbuf(), and rdstate().
Referenced by std::basic_ios< char_type, traits_type >::exceptions(), std::__detail::operator>>(), std::basic_istream< _CharT, _Traits >
rdbuf(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char_type, traits
and std::basic_istream< _CharT, _Traits >::unget().

copyfmt()

```
template<typename _CharT, typename _Traits>
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
            const basic_ios< _CharT, _Traits > & __rhs)  [inherited]
```

Copies fields of __rhs into this.

Parameters

__rhs	The source values for the copies.

Returns

Reference to this object.

All fields of __rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase_event. After copying, each (new) callback is invoked with copyfmt_event. The final step is to copy exceptions().
References basic_ios(), std::__addressof(), exceptions(), fill(), std::ios_base::flags(), std::ios_base::getloc(), std::ios_base::precision(), tie(), std::tie(), and std::ios_base::width().

eof()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

exceptions() [1/2]

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions () const  [inline], [nodiscard], [inherited]
```

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.
Referenced by clear(), and copyfmt().

exceptions() [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
            iostate __except)  [inline], [inherited]
```

Throwing exceptions on errors.

Parameters

__except	The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type std::ios_base::failure is thrown.
If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr « "Setting badbit\n";
```

```
    f.setstate (std::ios_base::badbit);

    std::cerr « "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.
Referenced by std::basic_ios< char_type, traits_type >::operator bool(), std::basic_ios< char_type, traits_type >::operator!(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >:: std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >:: and std::regex_traits< _Ch_type >::value().

fill() [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const  [inline], [nodiscard], [inherited]
```

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.
Referenced by copyfmt().

fill() [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
             char_type __ch)  [inline], [inherited]
```

Sets a new *empty* character.

Parameters

__ch	The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

flags() [1/2]

```
fmtflags std::ios_base::flags () const  [inline], [nodiscard], [inherited]
```

Access to format flags.

Returns

The format control flags for both input and output.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >
std::num_put< _CharT, _OutIter >::do_put(), std::num_put< _CharT, _OutIter >::do_put(), std::chrono::operator<<(),
std::operator<<(), std::__detail::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(),
std::operator>>(), std::operator>>(), and std::operator>>().

flags() [2/2]

```
fmtflags std::ios_base::flags (
            fmtflags __fmtfl)  [inline], [inherited]
```

Setting new format flags all at once.

Parameters

__fmtfl	The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *__fmtfl*.
References fmtflags.

flush()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::flush ()  [inherited]
```

Synchronizing the stream buffer.

Returns

*this

If `rdbuf()` is a null pointer, changes nothing.
Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit.
References std::ios_base::badbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits

gcount()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::gcount () const  [inline], [inherited]
```

Character counting.

Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

get() [1/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::get (
            void )  [inherited]
```

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().
References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

get() [2/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::get (
            __streambuf_type & __sb)  [inline], [inherited]
```

Extraction into another streambuf.

Parameters

__sb	A streambuf in which to store data.

Returns

*this

Returns get(__sb,widen('\n')).

get() [3/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
            __streambuf_type & __sb,
            char_type __delim)  [inherited]
```

Extraction into another streambuf.

Parameters

__sb	A streambuf in which to store data.
__delim	A "stop" character.

Returns

*this

Characters are extracted and inserted into *__sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals *__delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.
References _M_gcount, and std::ios_base::goodbit.

get() [4/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
            char_type & __c)  [inherited]
```

Simple extraction.

Parameters

_↩ _c	The character in which to store data.

Returns

*this

Tries to extract a character and store it in __c. If none are available, sets failbit and returns traits::eof().

Note

This function is not overloaded on signed char and unsigned char.

References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

get() [5/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::get (
            char_type * __s,
            streamsize __n)  [inline], [inherited]
```

Simple multiple-character extraction.

Parameters

_↩ _s	Pointer to an array.
_↩ _n	Maximum number of characters to store in *s*.

Returns

*this

Returns `get(__s,__n,widen('\n'))`.

get() [6/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
            char_type * __s,
            streamsize __n,
            char_type __delim)  [inherited]
```

Simple multiple-character extraction.

Parameters

__s	Pointer to an array.
__n	Maximum number of characters to store in __s.
__delim	A "stop" character.

Returns

*this

Characters are extracted and stored into __*s* until one of the following happens:

- __n-1 characters are stored
- the input sequence reaches EOF
- the next character equals __*delim*, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.
In any case, a null character is stored into the next location in the array.

Note

This function is not overloaded on signed char and unsigned char.

References _M_gcount, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_streambuf< _CharT, _Traits >:

getline() [1/3]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::getline (
            char_type * __s,
            streamsize __n)  [inline], [inherited]
```

String extraction.

Parameters

_↩ _*s*	A character array in which to store the data.
_↩ _*n*	Maximum number of characters to extract.

Returns

*this

Returns getline(__s,__n,widen('\n')).

getline() [2/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::getline (
            char_type * __s,
            streamsize __n,
            char_type __delim)  [inherited]
```

String extraction.

Parameters

__*s*	A character array in which to store the data.
__*n*	Maximum number of characters to extract.
__*delim*	A "stop" character.

Returns

*this

Extracts and stores characters into *__s* until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the *__s* array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)
In any case, a null character is stored in the next location in the array.
References _M_gcount, std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_streambuf< _CharT, _Traits >::sbumpc(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _Cha

getline() [3/3]

```
basic_istream< char > & std::basic_istream< char >::getline (
            char_type * __s,
            streamsize __n,
            char_type __delim)  [inherited]
```

Explicit specialization declarations, defined in src/istream.cc.

getloc()

```
locale std::ios_base::getloc () const  [inline], [nodiscard], [inherited]
```

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::money_put< _CharT, _OutIter >::do_put(), std::basic_ios< _CharT, _Tr std::chrono::operator<<(), std::operator>>(), std::operator>>(), and std::ws().

good()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if no error flags are set.

A wrapper around rdstate.
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::__detail::operator>>().

ignore() [1/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
            void )  [inherited]
```

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().
References _M_gcount, std::ios_base::goodbit, and std::basic_ios< _CharT, _Traits >::rdbuf().

ignore() [2/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
            streamsize __n)  [inherited]
```

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().
References _M_gcount, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_streambuf< _CharT, _Traits >:

ignore() [3/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
            streamsize __n,
            int_type __delim)  [inherited]
```

Discarding characters.

Parameters

__n	Number of characters to discard.
__delim	A "stop" character.

Returns

*this

Extracts characters and throws them away until one of the following happens:

- if *__n* != `std::numeric_limits<int>::max()`, *__n* characters are extracted
- the input sequence reaches end-of-file
- the next character equals *__delim* (in this case, the character is extracted); note that this condition will never occur if *__delim* equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.
References _M_gcount, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_streambuf< _CharT, _Traits >:

imbue()

```
template<typename _CharT, typename _Traits>
locale std::basic_ios< _CharT, _Traits >::imbue (
            const locale & __loc)  [inherited]
```

Moves to a new locale.

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.
Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html
References std::ios_base::getloc(), std::ios_base::imbue(), and rdbuf().
Referenced by std::chrono::operator<<().

init()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::init (
            basic_streambuf< _CharT, _Traits > * __sb)  [protected], [inherited]
```

All setup is performed here.
This is called from the public constructor. It is not virtual and cannot be redefined.
Referenced by std::basic_ios< char_type, traits_type >::basic_ios(), and std::basic_ostream< char_type, traits_type >::basic_ostream().

iword()

```
long & std::ios_base::iword (
            int __ix)  [inline], [inherited]
```

Access to integer array.

Parameters

__ix	Index into the array.

Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.
References iword().
Referenced by iword().

narrow()

```
template<typename _CharT, typename _Traits>
char std::basic_ios< _CharT, _Traits >::narrow (
            char_type __c,
            char __dfault) const  [inline], [inherited]
```

Squeezes characters.

Parameters

__c	The character to narrow.
__dfault	The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.
Returns the result of
`std::use_facet<ctype<char_type> >(getloc()).narrow(c,dfault)`
Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html

operator bool()

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool () const  [inline], [explicit], [nodiscard],
[inherited]
```

The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

operator"!()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator!  () const  [inline], [nodiscard], [inherited]
```

The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

operator<<() [1/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            __ios_type &(* __pf )(__ios_type &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

operator<<() [2/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            __ostream_type &(* __pf )(__ostream_type &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

operator<<() [3/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
            __streambuf_type * __sb)  [inherited]
```

Extracting from another streambuf.

Parameters

__sb	A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.
If __sb is NULL, the stream will set failbit in its error state.
Characters are extracted from __sb and inserted into *this until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from __sb, which sets failbit in the error state

If the function inserts no characters, failbit is set.

operator<<() [4/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            bool __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [5/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            const void * __p)  [inline], [inherited]
```

Pointer arithmetic inserters.

Parameters

_↩ _p	A variable of pointer type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [6/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            double __f)  [inline], [inherited]
```

Floating point arithmetic inserters.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [7/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            float __f)  [inline], [inherited]
```

Floating point arithmetic inserters.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [8/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
            int __n)  [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ *_n*	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [9/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            ios_base &(* __pf )(ios_base &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

operator<<() [10/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            long __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _*n*	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [11/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            long double __f)  [inline], [inherited]
```

Floating point arithmetic inserters.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [12/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            long long __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [13/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
            short __n)  [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.
References basic_ostream(), std::ios_base::badbit, std::ostreambuf_iterator< _CharT, _Traits >::failed(), std::ios_base::goodbit, std::num_put< _CharT, _OutIter >::put(), std::basic_ios< _CharT, _Traits >::setstate(), and std::use_facet().

operator<<() [14/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned int __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [15/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned long __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [16/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned long long __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [17/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned short __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator>>() [1/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            __ios_type &(* __pf )(__ios_type &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as std::ws and std::dec use these functions in constructs like std::cin >> std::ws. For more information, see the iomanip header.

operator>>() [2/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            __istream_type &(* __pf )(__istream_type &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the iomanip header.

operator>>() [3/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
            __streambuf_type * __sb)  [inherited]
```

Extracting into another streambuf.

Parameters

__sb	A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.
If `__sb` is NULL, the stream will set failbit in its error state.
Characters are extracted from this stream and inserted into the `__sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.
References std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

operator>>() [4/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            bool & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

operator>>() [5/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            double & __f)  [inline], [inherited]
```

Floating point arithmetic extractors.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [6/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            float & __f)  [inline], [inherited]
```

Floating point arithmetic extractors.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [7/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
            int & __n)  [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ *_n*	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.
References std::ios_base::badbit, std::ios_base::failbit, std::num_get< _CharT, _InIter >::get(), std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::setstate(), and std::use_facet().

operator>>() [8/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            ios_base &(* __pf )(ios_base &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the iomanip header.

operator>>() [9/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            long & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _*n*	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

operator>>() [10/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            long double & __f)  [inline], [inherited]
```

Floating point arithmetic extractors.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

operator>>() [11/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            long long & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [12/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
            short & __n)  [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.
References std::ios_base::badbit, std::ios_base::failbit, std::num_get< _CharT, _InIter >::get(), std::ios_base::goodbit, and std::use_facet().

operator>>() [13/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned int & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [14/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned long & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [15/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned long long & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [16/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned short & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [17/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            void *& __p)  [inline], [inherited]
```

Basic arithmetic extractors.

Parameters

_↩ _p	A variable of pointer type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

peek()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::peek (
            void )  [inherited]
```

Looking ahead in the stream.

Returns

The next character, or eof().

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

precision() [1/2]

```
streamsize std::ios_base::precision () const  [inline], [nodiscard], [inherited]
```

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::chrono::operator<<().

precision() [2/2]

```
streamsize std::ios_base::precision (
            streamsize __prec)  [inline], [inherited]
```

Changing flags.

Parameters

__prec	The new precision value.

Returns

The previous value of precision().

put()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (
            char_type __c)  [inherited]
```

Simple insertion.

Parameters

_↩ _c	The character to insert.

Returns

*this

Tries to insert __c.

Note

This function is not overloaded on signed char and unsigned char.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits

putback()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::putback (
            char_type __c)  [inherited]
```

Unextracting a single character.

Parameters

_↩ _c	The character to push back into the input stream.

Returns

*this

If rdbuf() is not null, calls rdbuf()->sputbackc(c).
If rdbuf() is null or if sputbackc() fails, sets badbit in the error state.

Note

This function first clears eofbit. Since no characters are extracted, the next call to gcount() will return 0, as required by DR 60.

References _M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::rdstate(), std::basic_ios< _CharT, _Traits >::setstate(), and std::basic_streambuf< _CharT, _Traits >::sputbackc().

pword()

```
void *& std::ios_base::pword (
            int __ix)  [inline], [inherited]
```

Access to void pointer array.

Parameters

_↩ _ix	Index into the array.

Returns

A reference to a void* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

rdbuf() [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
            basic_streambuf< _CharT, _Traits > * __sb)  [inherited]
```

Changing the underlying buffer.

Parameters

__sb	The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.
Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream     foo;         // or some other derived type
std::streambuf*  p = .....;

foo.ios::rdbuf(p);            // ios == basic_ios<char>
```

References clear().

rdbuf() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
__stringbuf_type * std::basic_stringstream< _CharT, _Traits, _Alloc >::rdbuf () const  [inline],
[nodiscard]
```

Accessing the underlying buffer.

Returns

The current basic_stringbuf buffer.

This hides both signatures of std::basic_ios::rdbuf().

rdstate()

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::rdstate () const  [inline], [nodiscard], [inherited]
```

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See std::ios_base::iostate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().
Referenced by std::basic_ios< char_type, traits_type >::bad(), clear(), std::basic_ios< char_type, traits_type >::eof(), std::basic_ios< char_type, traits_type >::fail(), std::basic_ios< char_type, traits_type >::good(), std::basic_istream< _CharT, _Traits >::p std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char_type, traits_type > and std::basic_istream< _CharT, _Traits >::unget().

read()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::read (
            char_type * __s,
            streamsize __n)  [inherited]
```

Extraction without delimiters.

Parameters

_↩ _s	A character array.
_↩ _n	Maximum number of characters to store.

Returns

*this

If the stream state is `good()`, extracts characters and stores them into *__s* until one of the following happens:

- *__n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

Note

This function is not overloaded on signed char and unsigned char.

References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

readsome()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::readsome (
            char_type * __s,
            streamsize __n)  [inherited]
```

Extraction until the buffer is exhausted, but no more.

Parameters

_↩ _s	A character array.
_↩ _n	Maximum number of characters to store.

Returns

The number of characters extracted.

Extracts characters and stores them into *__s* depending on the number of characters remaining in the streambuf's buffer, `rdbuf()->in_avail()`, called `A` here:

- if `A == -1`, sets eofbit and extracts no characters
- if `A == 0`, extracts no characters

- if A > 0, extracts min(A,n)

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.
References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::min(), std::basic_ios< _CharT, _Traits >::
and std::basic_ios< _CharT, _Traits >::setstate().

register_callback()

```
void std::ios_base::register_callback (
            event_callback __fn,
            int __index)  [inherited]
```

Add the callback __fn with parameter __index.

Parameters

__fn	The function to add.
__index	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.
References fmtflags.

seekg() [1/2]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
            off_type __off,
            ios_base::seekdir __dir)  [inherited]
```

Changing the current read position.

Parameters

__off	A file offset object.
__dir	The direction in which to seek.

Returns

*this

If fail() is not true, calls rdbuf()->pubseekoff(__off,__dir). If that function fails, sets failbit.

Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to gcount().

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::basic_ios< _CharT, _Traits >::fail(
std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::in, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits
and std::basic_ios< _CharT, _Traits >::setstate().

seekg() [2/2]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
            pos_type __pos)  [inherited]
```

Changing the current read position.

Parameters

__pos	A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets failbit.

Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::basic_ios< _CharT, _Traits >::fail(
std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::in, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits
and std::basic_ios< _CharT, _Traits >::setstate().

seekp() [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
            off_type __off,
            ios_base::seekdir __dir)  [inherited]
```

Changing the current write position.

Parameters

__off	A file offset object.
__dir	The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off,dir)`. If that function fails, sets failbit.
References std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::out, std::basic_ios< _CharT, _Traits >::rdbuf(),
and std::basic_ios< _CharT, _Traits >::setstate().

seekp() [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
            pos_type __pos)  [inherited]
```

Changing the current write position.

Parameters

__pos	A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.
References std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::out, std::basic_ios< _CharT, _Traits >::rdbuf(),
and std::basic_ios< _CharT, _Traits >::setstate().

setf() [1/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl)  [inline], [inherited]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.
References fmtflags.
Referenced by std::__detail::operator>>().

setf() [2/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl,
            fmtflags __mask)  [inline], [inherited]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.
__mask	The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.
References fmtflags.

setstate()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
            iostate __state)  [inline], [inherited]
```

Sets additional flags in the error state.

Parameters

__state	The additional state flag(s) to set.

See std::ios_base::iostate for the possible bit values.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::getline(), std::getline(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _Cha
std::operator>>(), std::operator>>(), std::operator>>(), std::tr2::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >:
std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >
std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::sync(), and std::ws().

str() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
__string_type std::basic_stringstream< _CharT, _Traits, _Alloc >::str () const &  [inline], [nodiscard]
```

Copying out the string buffer.

Returns

`rdbuf()->str()`

str() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_stringstream< _CharT, _Traits, _Alloc >::str (
            const __string_type & __s)  [inline]
```

Setting a new buffer.

Parameters

__s	The string to use as a new sequence.

Calls `rdbuf()->str(s)`.

sync()

```
template<typename _CharT, typename _Traits>
int std::basic_istream< _CharT, _Traits >::sync (
            void )  [inherited]
```

Synchronizing the stream buffer.

Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.
Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit and returns -1.
Otherwise, returns 0.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_streambuf< _CharT, _Traits >::pubsync(), std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

sync_with_stdio()

```
static bool std::ios_base::sync_with_stdio (
            bool __sync = true)  [static], [inherited]
```

Interaction with the standard C I/O objects.

Parameters

__sync	Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary

tellg()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::pos_type std::basic_istream< _CharT, _Traits >::tellg (
            void )  [inherited]
```

Getting the current read position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type`(-1) to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0,cur,in)`.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with putback, unget and seekg, eofbit is not cleared first.

References std::ios_base::badbit, std::ios_base::cur, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::in, and std::basic_ios< _CharT, _Traits >::rdbuf().

tellp()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits >::pos_type std::basic_ostream< _CharT, _Traits >::tellp ()  [inherited]
```

Getting the current write position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type`(-1) to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0,cur,out)`.
References std::ios_base::cur, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::out, and std::basic_ios< _CharT, _Traits >::rdbuf()

tie() [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie () const  [inline],
[nodiscard], [inherited]
```

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and copyfmt().

tie() [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie (
            basic_ostream< _CharT, _Traits > * __tiestr)  [inline], [inherited]
```

Ties this stream to an output stream.

Parameters

__tiestr	The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

unget()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::unget (
            void )  [inherited]
```

Unextracting the previous character.

Returns

*this

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.
If `rdbuf()` is null or if `sungetc()` fails, sets badbit in the error state.

Note

This function first clears eofbit. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References _M_gcount, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::ios_base::goodbit, and std::basic_ios< _CharT, _Traits >::rdstate().
Referenced by std::__detail::operator>>().

unsetf()

```
void std::ios_base::unsetf (
            fmtflags __mask)  [inline], [inherited]
```

Clearing format flags.

Parameters

__mask	The flags to unset.

This function clears *__mask* in the format flags.
References fmtflags.

widen()

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
            char __c) const  [inline], [inherited]
```

Widens characters.

Parameters

_↩ _c	The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.
Returns the result of
`std::use_facet<ctype<char_type> >(getloc()).widen(c)`
Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html
Referenced by std::basic_ios< char_type, traits_type >::fill(), std::getline(), std::getline(), and std::tr2::operator>>().

width() [1/2]

```
streamsize std::ios_base::width () const  [inline], [nodiscard], [inherited]
```

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_put< _CharT, _OutIter >::do_put(), std::operator>>(), and std::operator>>().

width() [2/2]

```
streamsize std::ios_base::width (
            streamsize __wide)  [inline], [inherited]
```

Changing flags.

Parameters

__wide	The new width value.

Returns

The previous value of width().

write()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::write (
            const char_type * __s,
            streamsize __n)  [inherited]
```

Character string insertion.

Parameters

_↩ _s	The array to insert.
_↩ _n	Maximum number of characters to insert.

Returns

*this

Characters are copied from __s and inserted into the stream until one of the following happens:

- __n characters are inserted
- inserting into the output sequence fails (in this case, badbit will be set in the stream's error state)

Note

This function is not overloaded on signed char and unsigned char.

xalloc()

```
static int std::ios_base::xalloc () throw ( )   [static], [inherited]
```

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the iword and pword functions. The expectation is that an application calls xalloc in order to obtain an index in the iword and pword arrays that can be used without fear of conflict.
The implementation maintains a static variable that is incremented and returned on each invocation. xalloc is guaranteed to return an index that is safe to use in the iword and pword arrays.

5.238.6 Member Data Documentation

_M_gcount

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::_M_gcount  [protected], [inherited]
```

The number of characters extracted in the previous unformatted function; see gcount().
Referenced by get(), get(), get(), get(), getline(), ignore(), ignore(), ignore(), putback(), read(), readsome(), and unget().

adjustfield

```
const fmtflags std::ios_base::adjustfield  [static], [inherited]
```

A mask of left|right|internal. Useful for the 2-arg form of setf.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::internal(), std::left(), and std::right().

app

```
const openmode std::ios_base::app  [static], [inherited]
```

Seek to end before each write.
Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

ate

```
const openmode std::ios_base::ate  [static], [inherited]
```

Open and seek to end immediately after opening.
Referenced by std::basic_filebuf< _CharT, _Traits >::open().

badbit

```
const iostate std::ios_base::badbit  [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< char_type, traits_type >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::basic_ios< _CharT, _Traits >::clear(), std::basic_ostream< _CharT, _Trai std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_ostream< _CharT, _Traits >::opera std::basic_ostream< char_type, traits_type >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >: std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< char_type, traits_ty std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_istream< char_type, traits_type and std::ws().

basefield

```
const fmtflags std::ios_base::basefield  [static], [inherited]
```

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.
Referenced by std::dec(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::hex(), and std::oct().

beg

```
const seekdir std::ios_base::beg  [static], [inherited]
```

Request a seek relative to the beginning of the stream.
Referenced by std::basic_filebuf< _CharT, _Traits >::seekpos().

binary

```
const openmode std::ios_base::binary  [static], [inherited]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.↩ filestreams.binary.
Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

boolalpha

```
const fmtflags std::ios_base::boolalpha  [static], [inherited]
```

Insert/extract `bool` in alphabetic rather than numeric format.
Referenced by std::boolalpha(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::noboolalpha().

cur

```
const seekdir std::ios_base::cur  [static], [inherited]
```

Request a seek relative to the current position within the sequence.
Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

dec

```
const fmtflags std::ios_base::dec  [static], [inherited]
```

Converts integer input or generates integer output in decimal base.
Referenced by std::dec().

end

```
const seekdir std::ios_base::end  [static], [inherited]
```

Request a seek relative to the current end of the sequence.
Referenced by std::basic_filebuf< _CharT, _Traits >::open(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff().

eofbit

```
const iostate std::ios_base::eofbit  [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.
Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >:
std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(),
std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter
std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::ge
std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< char_type, traits_type >::g
std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::putback(),
std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< char_type, traits_type >::read(), std::basic_istream< _CharT, _Traits
std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< char_type, traits_ty
std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

failbit

```
const iostate std::ios_base::failbit  [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::num_get< _CharT, _InIter >::do_get(),
std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(),
std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::ge
std::basic_istream< char_type, traits_type >::get(), std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::oper
std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(),
std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::s
std::basic_ostream< _CharT, _Traits >::seekp(), and std::basic_ostream< _CharT, _Traits >::seekp().

fixed

```
const fmtflags std::ios_base::fixed  [static], [inherited]
```

Generate floating-point output in fixed-point notation.
Referenced by std::fixed(), and std::hexfloat().

floatfield

```
const fmtflags std::ios_base::floatfield  [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.
Referenced by std::defaultfloat(), std::fixed(), std::hexfloat(), and std::scientific().

goodbit

```
const iostate std::ios_base::goodbit  [static], [inherited]
```

Indicates all is well.
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::num_get< _CharT, _InIter >::do_get(),
std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::d
std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >:
std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(),
std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore()
std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ostream< _CharT, _Traits >

std::basic_ostream< char_type, traits_type >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< char_type, traits_type >::seekg(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits and std::ws().

hex

const fmtflags std::ios_base::hex [static], [inherited]

Converts integer input or generates integer output in hexadecimal base.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::hex().

in

const openmode std::ios_base::in [static], [inherited]

Open for input. Default for `ifstream` and fstream.

Referenced by std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_stringbuf< _CharT, _Traits, _ std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_filebuf< _CharT, _Traits >::underflow(), std::basic_stringbuf< _CharT, _Traits, and std::basic_filebuf< _CharT, _Traits >::xsgetn().

internal

const fmtflags std::ios_base::internal [static], [inherited]

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Referenced by std::internal().

left

const fmtflags std::ios_base::left [static], [inherited]

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

Referenced by std::num_put< _CharT, _OutIter >::do_put(), and std::left().

oct

const fmtflags std::ios_base::oct [static], [inherited]

Converts integer input or generates integer output in octal base.

Referenced by std::oct().

out

const openmode std::ios_base::out [static], [inherited]

Open for output. Default for `ofstream` and fstream.

Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_stringbuf< _CharT, _Traits, std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

right

const fmtflags std::ios_base::right [static], [inherited]

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)
Referenced by std::right().

scientific

const fmtflags std::ios_base::scientific [static], [inherited]
Generates floating-point output in scientific notation.
Referenced by std::hexfloat(), and std::scientific().

showbase

const fmtflags std::ios_base::showbase [static], [inherited]
Generates a prefix indicating the numeric base of generated integer output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::noshowbase(), and std::showbase().

showpoint

const fmtflags std::ios_base::showpoint [static], [inherited]
Generates a decimal-point character unconditionally in generated floating-point output.
Referenced by std::noshowpoint(), and std::showpoint().

showpos

const fmtflags std::ios_base::showpos [static], [inherited]
Generates a + sign in non-negative generated numeric output.
Referenced by std::noshowpos(), and std::showpos().

skipws

const fmtflags std::ios_base::skipws [static], [inherited]
Skips leading white space before certain input operations.
Referenced by std::noskipws(), std::__detail::operator>>(), and std::skipws().

trunc

const openmode std::ios_base::trunc [static], [inherited]
Truncate an existing stream when opening. Default for `ofstream`.

unitbuf

const fmtflags std::ios_base::unitbuf [static], [inherited]
Flushes output after each output operation.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::nounitbuf(), and std::unitbuf().

uppercase

const fmtflags std::ios_base::uppercase [static], [inherited]
Replaces certain lowercase letters with their uppercase equivalents in generated output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::nouppercase(), and std::uppercase().
The documentation for this class was generated from the following files:

- iosfwd
- sstream

5.239 std::bernoulli_distribution Class Reference

`#include <random>`

Classes

- struct param_type

Public Types

- typedef bool result_type

Public Member Functions

- bernoulli_distribution ()
- **bernoulli_distribution** (const param_type &__p)
- bernoulli_distribution (double __p)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- double p () const
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- bool operator== (const bernoulli_distribution &__d1, const bernoulli_distribution &__d2)

5.239.1 Detailed Description

A Bernoulli random number distribution.
Generates a sequence of true and false values with likelihood p that true will come up and $(1 - p)$ that false will appear.

Since

C++11

5.239.2 Member Typedef Documentation

result_type

`typedef bool std::bernoulli_distribution::result_type`
The type of the range of the distribution.

5.239.3 Constructor & Destructor Documentation

bernoulli_distribution() [1/2]

```
std::bernoulli_distribution::bernoulli_distribution ()  [inline]
```

Constructs a Bernoulli distribution with likelihood 0.5.
References bernoulli_distribution().
Referenced by bernoulli_distribution(), and operator==.

bernoulli_distribution() [2/2]

```
std::bernoulli_distribution::bernoulli_distribution (
            double __p)  [inline], [explicit]
```

Constructs a Bernoulli distribution with likelihood p.

Parameters

_↩ _p	[IN] The likelihood of a true result being returned. Must be in the interval $[0, 1]$.

5.239.4 Member Function Documentation

max()

```
result_type std::bernoulli_distribution::max () const  [inline]
```

Returns the least upper bound value of the distribution.
References std::numeric_limits< _Tp >::max().

min()

```
result_type std::bernoulli_distribution::min () const  [inline]
```

Returns the greatest lower bound value of the distribution.
References std::numeric_limits< _Tp >::min().

operator()()

```
template<typename _UniformRandomNumberGenerator>
result_type std::bernoulli_distribution::operator() (
            _UniformRandomNumberGenerator & __urng)  [inline]
```

Generating functions.
References operator()().
Referenced by operator()().

p()

```
double std::bernoulli_distribution::p () const  [inline]
```

Returns the p parameter of the distribution.

param() [1/2]

```
param_type std::bernoulli_distribution::param () const  [inline]
```

Returns the parameter set of the distribution.
Referenced by std::operator>>().

param() [2/2]

```
void std::bernoulli_distribution::param (
            const param_type & __param)  [inline]
```

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

reset()

```
void std::bernoulli_distribution::reset ()  [inline]
```

Resets the distribution state.
Does nothing for a Bernoulli distribution.

5.239.5 Friends And Related Symbol Documentation

operator==

```
bool operator== (
            const bernoulli_distribution & __d1,
            const bernoulli_distribution & __d2)  [friend]
```

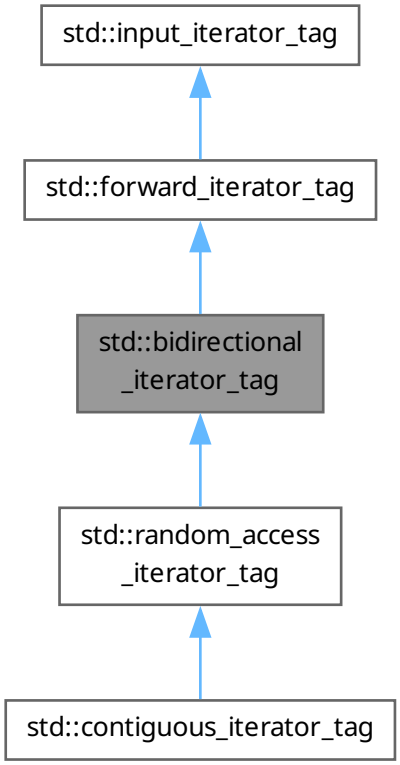
Return true if two Bernoulli distributions have the same parameters.
References bernoulli_distribution().
The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

5.240 std::bidirectional_iterator_tag Struct Reference

```
#include <stl_iterator_base_types.h>
```

Inheritance diagram for std::bidirectional_iterator_tag:



5.240.1 Detailed Description

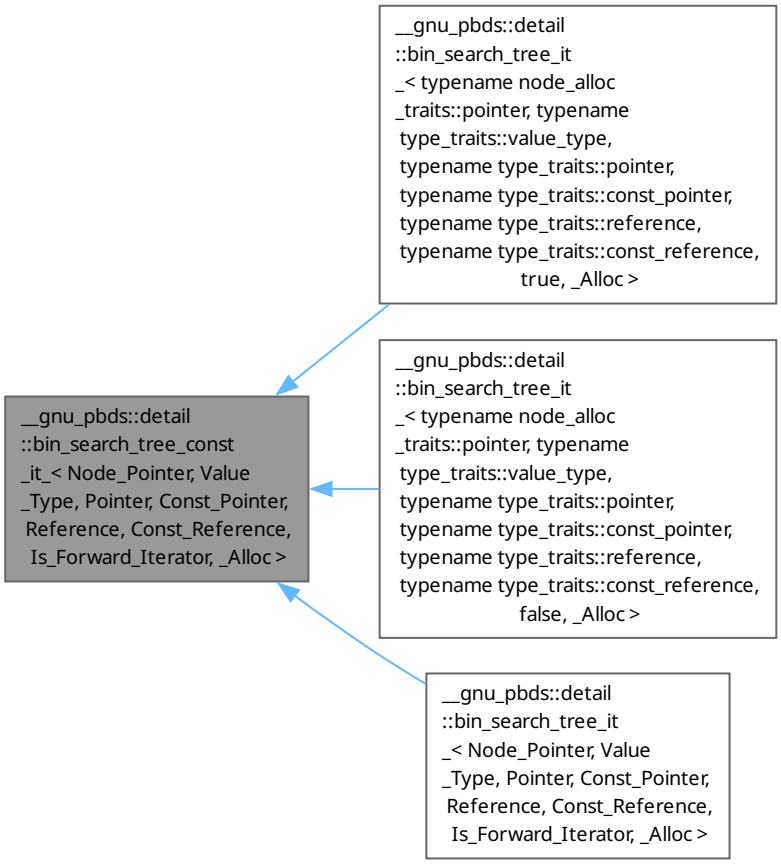
Bidirectional iterators support a superset of forward iterator operations.
The documentation for this struct was generated from the following file:

- stl_iterator_base_types.h

5.241 __gnu_pbds::detail::bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > Class Template Reference

`#include <point_iterators.hpp>`
Inheritance diagram for __gnu_pbds::detail::bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_↩

Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >:



Public Types

- typedef Const_Pointer **const_pointer**
- typedef Const_Reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef std::bidirectional_iterator_tag **iterator_category**
- typedef Pointer **pointer**
- typedef Reference **reference**
- typedef Value_Type **value_type**

Public Member Functions

- **bin_search_tree_const_it_** (const bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_↩ Pointer, Reference, Const_Reference, !Is_Forward_Iterator, _Alloc > &other)
- **bin_search_tree_const_it_** (const Node_Pointer p_nd=0)
- bool **operator!=** (const bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, !Is_Forward_Iterator, _Alloc > &other) const

- bool **operator!=** (const bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > &other) const
- const_reference **operator*** () const
- bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > & **operator++** ()
- bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > **operator++** (int)
- bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > & **operator--** ()
- bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > **operator--** (int)
- const_pointer **operator->** () const
- bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > & **operator=** (const bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, !Is_Forward_Iterator, _Alloc > &other)
- bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > & **operator=** (const bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > &other)
- bool **operator==** (const bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, !Is_Forward_Iterator, _Alloc > &other) const
- bool **operator==** (const bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > &other) const

Public Attributes

- Node_Pointer **m_p_nd**

Protected Member Functions

- void **dec** (false_type)
- void **dec** (true_type)
- void **inc** (false_type)
- void **inc** (true_type)

5.241.1 Detailed Description

template<typename Node_Pointer, typename Value_Type, typename Pointer, typename Const_Pointer, typename Reference, typename Const_Reference, bool Is_Forward_Iterator, typename _Alloc>
class __gnu_pbds::detail::bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >

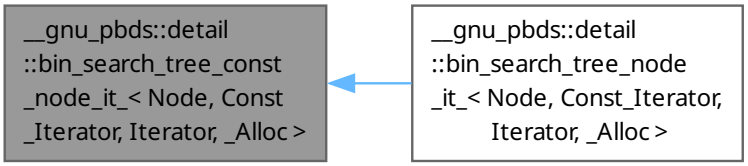
Const iterator.
The documentation for this class was generated from the following file:

- point_iterators.hpp

5.242 __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > Class Template Reference

`#include <node_iterators.hpp>`

Inheritance diagram for __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >:



Public Types

- typedef Const_Iterator const_reference
- typedef trivial_iterator_difference_type difference_type
- typedef trivial_iterator_tag iterator_category
- typedef rebind_traits< _Alloc, metadata_type >::const_reference metadata_const_reference
- typedef Node::metadata_type metadata_type
- typedef Const_Iterator reference
- typedef Const_Iterator value_type

Public Member Functions

- **bin_search_tree_const_node_it_** (const node_pointer p_nd=0)
- bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > get_l_child () const
- metadata_const_reference get_metadata () const
- bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > get_r_child () const
- bool operator!= (const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > &other) const
- const_reference operator* () const
- bool operator== (const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > &other) const

Public Attributes

- node_pointer **m_p_nd**

5.242.1 Detailed Description

template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
class __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >

Const node iterator.

5.242.2 Member Typedef Documentation

const_reference

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef Const_Iterator __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator,
Iterator, _Alloc >::const_reference
```

Iterator's __const reference type.

difference_type

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef trivial_iterator_difference_type __gnu_pbds::detail::bin_search_tree_const_node_it_<
Node, Const_Iterator, Iterator, _Alloc >::difference_type
```
Difference type.

iterator_category

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef trivial_iterator_tag __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_↩
Iterator, Iterator, _Alloc >::iterator_category
```
Category.

metadata_const_reference

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef rebind_traits<_Alloc,metadata_type>::const_reference __gnu_pbds::detail::bin_search_tree_const_node_it_<
Node, Const_Iterator, Iterator, _Alloc >::metadata_const_reference
```
Const metadata reference type.

metadata_type

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef Node::metadata_type __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_↩
Iterator, Iterator, _Alloc >::metadata_type
```
Metadata type.

reference

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef Const_Iterator __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator,
Iterator, _Alloc >::reference
```
Iterator's reference type.

value_type

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef Const_Iterator __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator,
Iterator, _Alloc >::value_type
```
Iterator's value type.

5.242.3 Member Function Documentation

get_l_child()

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > __gnu_pbds::detail::bin_search_tree_cons
Node, Const_Iterator, Iterator, _Alloc >::get_l_child () const  [inline]
```
Returns the __const node iterator associated with the left node.

get_metadata()

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
metadata_const_reference __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator,
Iterator, _Alloc >::get_metadata () const  [inline]
```

Metadata access.

get_r_child()

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > __gnu_pbds::detail::bin_search_tree_cons
Node, Const_Iterator, Iterator, _Alloc >::get_r_child () const  [inline]
```

Returns the __const node iterator associated with the right node.

operator"!=()

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
bool __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc
>::operator!= (
            const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > &
other) const  [inline]
```

Compares (negatively) to a different iterator object.

operator*()

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
const_reference __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator,
_Alloc >::operator* () const  [inline]
```

Access.

operator==()

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
bool __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc
>::operator== (
            const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > &
other) const  [inline]
```

Compares to a different iterator object.

The documentation for this class was generated from the following file:

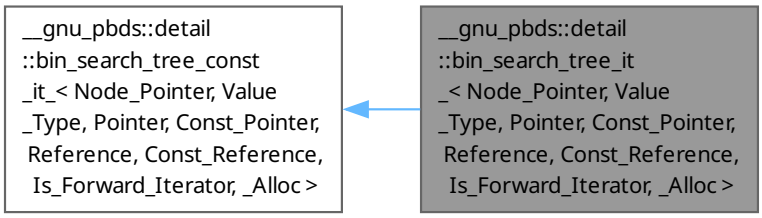
- bin_search_tree_/node_iterators.hpp

5.243 __gnu_pbds::detail::bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > Class Template Reference

```
#include <point_iterators.hpp>
```

Inheritance diagram for __gnu_pbds::detail::bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer,

Reference, Const_Reference, Is_Forward_Iterator, _Alloc >:



Public Types

- typedef Const_Pointer **const_pointer**
- typedef Const_Reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef std::bidirectional_iterator_tag **iterator_category**
- typedef Pointer **pointer**
- typedef Reference **reference**
- typedef Value_Type **value_type**

Public Member Functions

- **bin_search_tree_it_** (const bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, !Is_Forward_Iterator, _Alloc > &other)
- **bin_search_tree_it_** (const Node_Pointer p_nd=0)
- bool **operator!=** (const bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, !Is_Forward_Iterator, _Alloc > &other) const
- bool **operator!=** (const bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > &other) const
- bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >::reference **operator*** () const
- bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > & **operator++** ()
- bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > **operator++** (int)
- bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > & **operator--** ()
- bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > **operator--** (int)
- bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >::pointer **operator->** () const
- bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > & **operator=** (const bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, !Is_Forward_Iterator, _Alloc > &other)
- bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > & **operator=** (const bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > &other)

- bool **operator==** (const bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, !Is_Forward_Iterator, _Alloc > &other) const
- bool **operator==** (const bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > &other) const

Public Attributes

- Node_Pointer **m_p_nd**

Protected Types

- typedef bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_↩ Reference, Is_Forward_Iterator, _Alloc > **base_it_type**

Protected Member Functions

- void **dec** (false_type)
- void **dec** (true_type)
- void **inc** (false_type)
- void **inc** (true_type)

5.243.1 Detailed Description

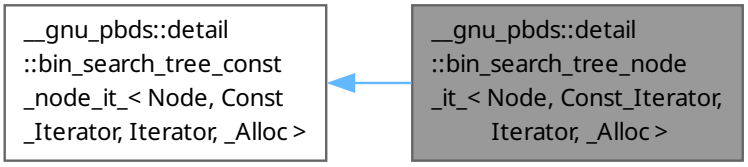
template<typename Node_Pointer, typename Value_Type, typename Pointer, typename Const_Pointer, typename Reference, typename Const_Reference, bool Is_Forward_Iterator, typename _Alloc>
class __gnu_pbds::detail::bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >

Iterator.
The documentation for this class was generated from the following file:

- point_iterators.hpp

5.244 __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc > Class Template Reference

`#include <node_iterators.hpp>`
Inheritance diagram for __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >:



Public Types

- typedef Iterator const_reference
- typedef trivial_iterator_difference_type difference_type
- typedef trivial_iterator_tag iterator_category
- typedef rebind_traits< _Alloc, metadata_type >::const_reference metadata_const_reference
- typedef Node::metadata_type metadata_type
- typedef Iterator reference
- typedef Iterator value_type

Public Member Functions

- **bin_search_tree_node_it_** (const node_pointer p_nd=0)
- bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc > get_l_child () const
- metadata_const_reference get_metadata () const
- bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc > get_r_child () const
- bool operator!= (const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > &other) const
- Iterator operator* () const
- bool operator== (const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > &other) const

Public Attributes

- node_pointer **m_p_nd**

5.244.1 Detailed Description

template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
class __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >

Node iterator.

5.244.2 Member Typedef Documentation

const_reference

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef Iterator __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _↩
Alloc >::const_reference
```

Iterator's __const reference type.

difference_type

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef trivial_iterator_difference_type __gnu_pbds::detail::bin_search_tree_const_node_it_<
Node, Const_Iterator, Iterator, _Alloc >::difference_type  [inherited]
```

Difference type.

iterator_category

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef trivial_iterator_tag __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_↩
Iterator, Iterator, _Alloc >::iterator_category  [inherited]
```

Category.

metadata_const_reference

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef rebind_traits<_Alloc,metadata_type>::const_reference __gnu_pbds::detail::bin_search_tree_const_node_it_<
Node, Const_Iterator, Iterator, _Alloc >::metadata_const_reference [inherited]
```

Const metadata reference type.

metadata_type

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef Node::metadata_type __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_↩
Iterator, Iterator, _Alloc >::metadata_type [inherited]
```

Metadata type.

reference

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef Iterator __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _↩
Alloc >::reference
```

Iterator's reference type.

value_type

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef Iterator __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _↩
Alloc >::value_type
```

Iterator's value type.

5.244.3 Member Function Documentation

get_l_child()

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc > __gnu_pbds::detail::bin_search_tree_node_it_<
Node, Const_Iterator, Iterator, _Alloc >::get_l_child () const [inline]
```

Returns the node iterator associated with the left node.

get_metadata()

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
metadata_const_reference __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator,
Iterator, _Alloc >::get_metadata () const [inline], [inherited]
```

Metadata access.

get_r_child()

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc > __gnu_pbds::detail::bin_search_tree_node_it_<
Node, Const_Iterator, Iterator, _Alloc >::get_r_child () const [inline]
```

Returns the node iterator associated with the right node.

operator"!=()

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
bool __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc
>::operator!= (
```

```
            const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > &
other) const  [inline], [inherited]
```

Compares (negatively) to a different iterator object.

operator*()

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
Iterator __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >↩
::operator* () const  [inline]
```

Access.

operator==()

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
bool __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc
>::operator== (
            const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > &
other) const  [inline], [inherited]
```

Compares to a different iterator object.

The documentation for this class was generated from the following file:

- bin_search_tree_/node_iterators.hpp

5.245 __gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc > Struct Template Reference

```
#include <traits.hpp>
```

Public Types

- typedef bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc > **const_reverse_iterator**
- typedef Node **node**
- typedef bin_search_tree_const_node_it_< Node, point_const_iterator, point_iterator, _Alloc > node_const_iterator
- typedef bin_search_tree_node_it_< Node, point_const_iterator, point_iterator, _Alloc > **node_iterator**
- typedef Node_Update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > **node_update**
- typedef __gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * **null_node**↩ **_update_pointer**
- typedef bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc > **point_const_iterator**
- typedef bin_search_tree_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc > **point_iterator**
- typedef bin_search_tree_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc > **reverse_iterator**

5.245.1 Detailed Description

template<typename Key, typename Mapped, class Cmp_Fn, template< typename Node_CItr, class Node_Itr, class _Cmp_Fn, typename _Alloc > class Node_Update, class Node, typename _Alloc>
struct __gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >

Binary search tree traits, primary template.

5.245.2 Member Typedef Documentation

node_const_iterator

```
template<typename Key, typename Mapped, class Cmp_Fn, template< typename Node_CItr, class Node_↩
Itr, class _Cmp_Fn, typename _Alloc > class Node_Update, class Node, typename _Alloc>
typedef bin_search_tree_const_node_it_< Node, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::b
Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >::node_const_iterator
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.
The documentation for this struct was generated from the following file:

- bin_search_tree_/traits.hpp

5.246 __gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc > Struct Template Reference

```
#include <traits.hpp>
```

Public Types

- typedef bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc > **const_reverse_iterator**
- typedef bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc > **const_reverse_iterator**
- typedef Node **node**
- typedef Node **node**
- typedef bin_search_tree_const_node_it_< Node, point_const_iterator, point_iterator, _Alloc > node_const_iterator
- typedef bin_search_tree_const_node_it_< Node, point_const_iterator, point_iterator, _Alloc > node_const_iterator
- typedef bin_search_tree_node_it_< Node, point_const_iterator, point_iterator, _Alloc > **node_iterator**
- typedef node_const_iterator **node_iterator**
- typedef Node_Update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > **node_update**
- typedef Node_Update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > **node_update**
- typedef __gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * **null_node↩ _update_pointer**
- typedef __gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * **null_node↩ _update_pointer**
- typedef bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc > **point_const_iterator**
- typedef bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc > **point_const_iterator**
- typedef bin_search_tree_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc > **point_iterator**
- typedef point_const_iterator **point_iterator**
- typedef bin_search_tree_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc > **reverse_iterator**
- typedef const_reverse_iterator **reverse_iterator**

5.246.1 Detailed Description

template<typename Key, class Cmp_Fn, template< typename Node_CItr, class Node_Itr, class _Cmp_Fn, typename _Alloc > class Node_Update, class Node, typename _Alloc>
struct __gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >

Specialization.

5.246.2 Member Typedef Documentation

node_const_iterator [1/2]

```
typedef bin_search_tree_const_node_it_< Node, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::b
Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >::node_const_iterator
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

node_const_iterator [2/2]

```
template<typename Key, class Cmp_Fn, template< typename Node_CItr, class Node_Itr, class _Cmp_Fn,
typename _Alloc > class Node_Update, class Node, typename _Alloc>
typedef bin_search_tree_const_node_it_< Node, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::b
Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >::node_const_iterator
```

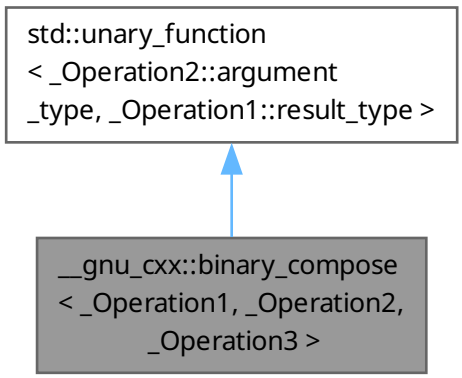
This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.
The documentation for this struct was generated from the following file:

- bin_search_tree_/traits.hpp

5.247 __gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 > Class Template Reference

```
#include <functional>
```

Inheritance diagram for __gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >:



Public Types

- typedef _Operation2::argument_type argument_type

- typedef _Operation1::result_type result_type

Public Member Functions

- **binary_compose** (const _Operation1 &__x, const _Operation2 &__y, const _Operation3 &__z)
- _Operation1::result_type **operator()** (const typename _Operation2::argument_type &__x) const

Protected Attributes

- _Operation1 **_M_fn1**
- _Operation2 **_M_fn2**
- _Operation3 **_M_fn3**

5.247.1 Detailed Description

template<class _Operation1, class _Operation2, class _Operation3>
class __gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >

An SGI extension .

5.247.2 Member Typedef Documentation

argument_type

```
typedef _Operation2::argument_type std::unary_function< _Operation2::argument_type, _Operation1↩
::result_type >::argument_type  [inherited]
```

argument_type is the type of the argument

result_type

```
typedef _Operation1::result_type std::unary_function< _Operation2::argument_type, _Operation1↩
::result_type >::result_type  [inherited]
```

result_type is the return type

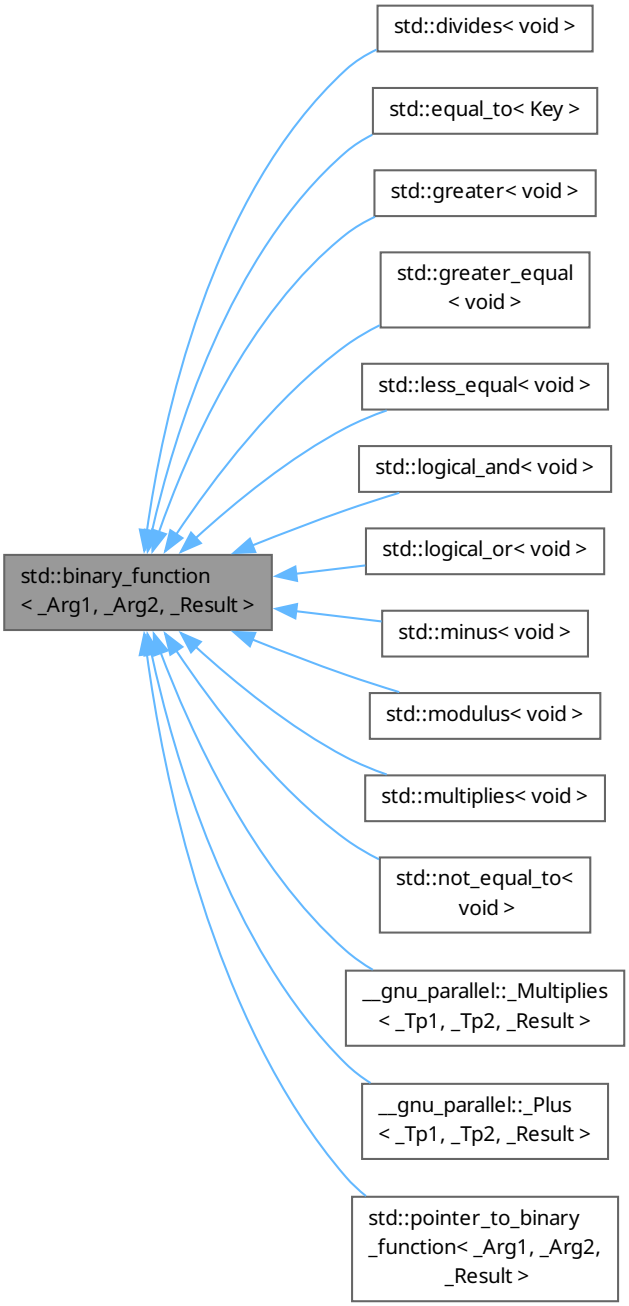
The documentation for this class was generated from the following file:

- ext/functional

5.248 std::binary_function< _Arg1, _Arg2, _Result > Struct Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::binary_function< _Arg1, _Arg2, _Result >:



Public Types

- typedef _Arg1 first_argument_type
- typedef _Result result_type
- typedef _Arg2 second_argument_type

5.248.1 Detailed Description

template<typename _Arg1, typename _Arg2, typename _Result>
struct std::binary_function< _Arg1, _Arg2, _Result >

Helper for defining adaptable binary function objects.

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

5.248.2 Member Typedef Documentation

first_argument_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result >::first_argument_type
```

`first_argument_type` is the type of the first argument

result_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Result std::binary_function< _Arg1, _Arg2, _Result >::result_type
```

`result_type` is the return type

second_argument_type

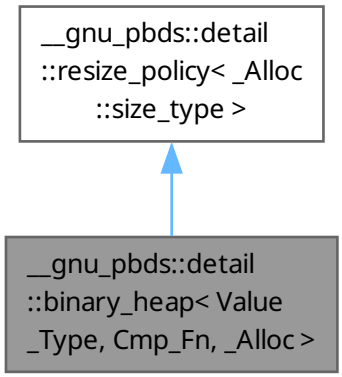
```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result >::second_argument_type
```

`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

5.249 __gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc > Class Template Reference

`#include <binary_heap_.hpp>`
Inheritance diagram for __gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >:


__gnu_pbds::detail
::resize_policy< _Alloc
::size_type >
__gnu_pbds::detail
::binary_heap< Value
_Type, Cmp_Fn, _Alloc >
__gnu_pbds::detail
::binary_heap< Value
_Type, Cmp_Fn, _Alloc >
__gnu_pbds::detail
::resize_policy< _Alloc
::size_type

Public Types

- typedef _Alloc **allocator_type**
- typedef Cmp_Fn **cmp_fn**
- typedef cond_dealtor< value_type, _Alloc > **cond_dealtor_t**
- typedef binary_heap_const_iterator_< value_type, entry, simple_value, _Alloc > **const_iterator**
- typedef __rebind_v::const_pointer **const_pointer**
- typedef __rebind_v::const_reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef __conditional_type< simple_value, value_type, pointer >::__type **entry**
- typedef rebind_traits< _Alloc, entry >::allocator_type **entry_allocator**
- typedef entry_cmp< Value_Type, Cmp_Fn, _Alloc, is_simple< Value_Type >::value >::type **entry_cmp**
- typedef rebind_traits< _Alloc, entry >::pointer **entry_pointer**
- typedef const_iterator **iterator**
- typedef binary_heap_point_const_iterator_< value_type, entry, simple_value, _Alloc > **point_const_iterator**
- typedef point_const_iterator **point_iterator**
- typedef __rebind_v::pointer **pointer**
- typedef __rebind_v::reference **reference**
- typedef __gnu_pbds::detail::resize_policy< typename _Alloc::size_type > **resize_policy**
- typedef _Alloc::size_type **size_type**
- typedef Value_Type **value_type**

Public Member Functions

- **binary_heap** (const binary_heap &)
- **binary_heap** (const cmp_fn &)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- void **erase** (point_iterator)
- void **erase_at** (entry_pointer, size_type, false_type)
- void **erase_at** (entry_pointer, size_type, true_type)
- template<typename Pred>
  size_type **erase_if** (Pred)
- Cmp_Fn & **get_cmp_fn** ()
- const Cmp_Fn & **get_cmp_fn** () const
- size_type **get_new_size_for_arbitrary** (size_type) const
- size_type **get_new_size_for_grow** () const
- size_type **get_new_size_for_shrink** () const
- bool **grow_needed** (size_type) const
- void **join** (binary_heap &)
- size_type **max_size** () const
- void **modify** (point_iterator, const_reference)
- void **notify_arbitrary** (size_type)
- void **notify_grow_resize** ()
- void **notify_shrink_resize** ()
- void **pop** ()
- point_iterator **push** (const_reference)
- bool **resize_needed_for_grow** (size_type) const

- bool **resize_needed_for_shrink** (size_type) const
- bool **shrink_needed** (size_type) const
- size_type **size** () const
- template<typename Pred>
  void **split** (Pred, binary_heap &)
- void **swap** (binary_heap &)
- void **swap** (resize_policy< _Alloc::size_type > &)
- const_reference **top** () const

Static Public Attributes

- static const _Alloc::size_type **min_size**

Protected Member Functions

- template<typename It>
  void **copy_from_range** (It, It)

5.249.1 Detailed Description

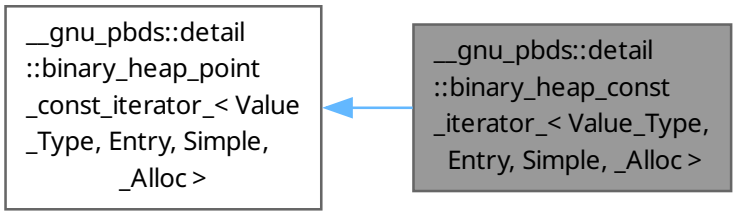
template<typename Value_Type, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >

Binary heaps composed of resize and compare policies.
Based on CLRS.
The documentation for this class was generated from the following file:

- binary_heap_.hpp

5.250 __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc > Class Template Reference

`#include <const_iterator.hpp>`
Inheritance diagram for __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >:



Public Types

- typedef base_type::const_pointer const_pointer
- typedef base_type::const_reference const_reference
- typedef _Alloc::difference_type difference_type

- typedef std::forward_iterator_tag iterator_category
- typedef base_type::pointer pointer
- typedef base_type::reference reference
- typedef base_type::value_type value_type

Public Member Functions

- binary_heap_const_iterator_ ()
- binary_heap_const_iterator_ (const binary_heap_const_iterator_ &other)
- **binary_heap_const_iterator_** (entry_pointer p_e)
- bool operator!= (const binary_heap_const_iterator_ &other) const
- bool operator!= (const binary_heap_point_const_iterator_ &other) const
- const_reference operator* () const
- binary_heap_const_iterator_ & **operator++** ()
- binary_heap_const_iterator_ **operator++** (int)
- const_pointer operator-> () const
- bool operator== (const binary_heap_const_iterator_ &other) const
- bool operator== (const binary_heap_point_const_iterator_ &other) const

Public Attributes

- entry_pointer **m_p_e**

5.250.1 Detailed Description

template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
class __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >

Const point-type iterator.

5.250.2 Member Typedef Documentation

const_pointer

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef base_type::const_pointer __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type,
Entry, Simple, _Alloc >::const_pointer
```

Iterator's const pointer type.

const_reference

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef base_type::const_reference __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type,
Entry, Simple, _Alloc >::const_reference
```

Iterator's const reference type.

difference_type

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef _Alloc::difference_type __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type,
Entry, Simple, _Alloc >::difference_type
```

Difference type.

iterator_category

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef std::forward_iterator_tag __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type,
Entry, Simple, _Alloc >::iterator_category
```
Category.

pointer

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef base_type::pointer __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry,
Simple, _Alloc >::pointer
```
Iterator's pointer type.

reference

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef base_type::reference __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry,
Simple, _Alloc >::reference
```
Iterator's reference type.

value_type

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef base_type::value_type __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry,
Simple, _Alloc >::value_type
```
Iterator's value type.

5.250.3 Constructor & Destructor Documentation

binary_heap_const_iterator_() [1/2]

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::binary_↩
heap_const_iterator_ ()  [inline]
```
Default constructor.

binary_heap_const_iterator_() [2/2]

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::binary_↩
heap_const_iterator_ (
            const binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other)
[inline]
```
Copy constructor.

5.250.4 Member Function Documentation

operator"!=() [1/2]

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
bool __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator!=
(
            const binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other)
const  [inline]
```
Compares content (negatively) to a different iterator object.

operator"!=() [2/2]

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
bool __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >↩
::operator!= (
            const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other)
const  [inline], [inherited]
```

Compares content (negatively) to a different iterator object.

operator*()

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
const_reference __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple,
_Alloc >::operator* () const  [inline], [inherited]
```

Access.

operator->()

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
const_pointer __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, ↩
_Alloc >::operator-> () const  [inline], [inherited]
```

Access.

operator==() [1/2]

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
bool __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::operator==
(
            const binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other)
const  [inline]
```

Compares content to a different iterator object.

operator==() [2/2]

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
bool __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >↩
::operator== (
            const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other)
const  [inline], [inherited]
```

Compares content to a different iterator object.

The documentation for this class was generated from the following file:

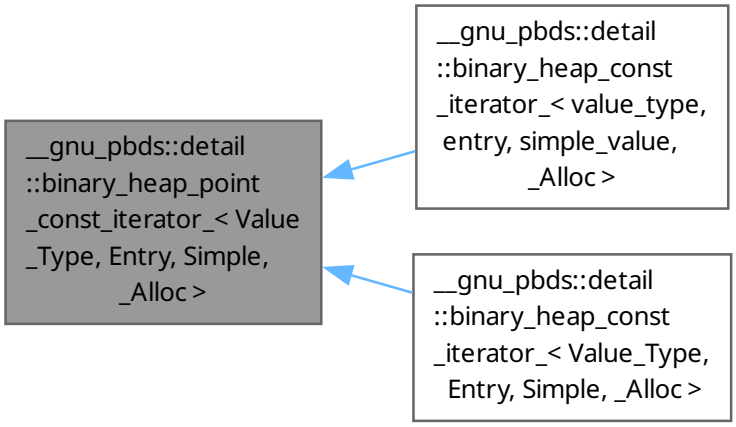
- binary_heap_/const_iterator.hpp

5.251 __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > Class Template Reference

```
#include <point_const_iterator.hpp>
```

Inheritance diagram for __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc

>:



Public Types

- typedef rebind_traits< _Alloc, value_type >::const_pointer const_pointer
- typedef rebind_traits< _Alloc, value_type >::const_reference const_reference
- typedef trivial_iterator_difference_type difference_type
- typedef trivial_iterator_tag iterator_category
- typedef rebind_traits< _Alloc, value_type >::pointer pointer
- typedef rebind_traits< _Alloc, value_type >::reference reference
- typedef Value_Type value_type

Public Member Functions

- binary_heap_point_const_iterator_ ()
- binary_heap_point_const_iterator_ (const binary_heap_point_const_iterator_ &other)
- **binary_heap_point_const_iterator_** (entry_pointer p_e)
- bool operator!= (const binary_heap_point_const_iterator_ &other) const
- const_reference operator* () const
- const_pointer operator-> () const
- bool operator== (const binary_heap_point_const_iterator_ &other) const

Public Attributes

- entry_pointer **m_p_e**

Protected Types

- typedef rebind_traits< _Alloc, Entry >::pointer **entry_pointer**

5.251.1 Detailed Description

template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
class __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >

Const point-type iterator.

5.251.2 Member Typedef Documentation

const_pointer

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef rebind_traits<_Alloc,value_type>::const_pointer __gnu_pbds::detail::binary_heap_point_const_iterator_<
Value_Type, Entry, Simple, _Alloc >::const_pointer
```

Iterator's const pointer type.

const_reference

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef rebind_traits<_Alloc,value_type>::const_reference __gnu_pbds::detail::binary_heap_point_const_iterator_<
Value_Type, Entry, Simple, _Alloc >::const_reference
```

Iterator's const reference type.

difference_type

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef trivial_iterator_difference_type __gnu_pbds::detail::binary_heap_point_const_iterator_<
Value_Type, Entry, Simple, _Alloc >::difference_type
```

Difference type.

iterator_category

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef trivial_iterator_tag __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type,
Entry, Simple, _Alloc >::iterator_category
```

Category.

pointer

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef rebind_traits<_Alloc,value_type>::pointer __gnu_pbds::detail::binary_heap_point_const_iterator_<
Value_Type, Entry, Simple, _Alloc >::pointer
```

Iterator's pointer type.

reference

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef rebind_traits<_Alloc,value_type>::reference __gnu_pbds::detail::binary_heap_point_const_iterator_<
Value_Type, Entry, Simple, _Alloc >::reference
```

Iterator's reference type.

value_type

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef Value_Type __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry,
Simple, _Alloc >::value_type
```

Iterator's value type.

5.251.3 Constructor & Destructor Documentation

binary_heap_point_const_iterator_() [1/2]

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >↩
::binary_heap_point_const_iterator_ ()  [inline]
```

Default constructor.

binary_heap_point_const_iterator_() [2/2]

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >↩
::binary_heap_point_const_iterator_ (
            const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other)
[inline]
```

Copy constructor.

5.251.4 Member Function Documentation

operator"!=()

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
bool __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >↩
::operator!= (
            const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other)
const  [inline]
```

Compares content (negatively) to a different iterator object.

operator*()

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
const_reference __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple,
_Alloc >::operator* () const  [inline]
```

Access.

operator->()

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
const_pointer __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, ↩
_Alloc >::operator-> () const  [inline]
```

Access.

operator==()

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
bool __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >↩
::operator== (
            const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other)
const  [inline]
```

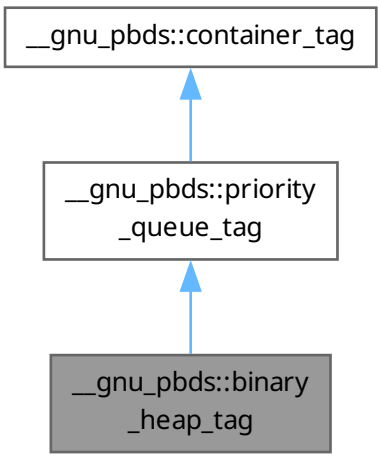
Compares content to a different iterator object.

The documentation for this class was generated from the following file:

- binary_heap_/point_const_iterator.hpp

5.252 __gnu_pbds::binary_heap_tag Struct Reference

`#include <tag_and_trait.hpp>`
Inheritance diagram for __gnu_pbds::binary_heap_tag:



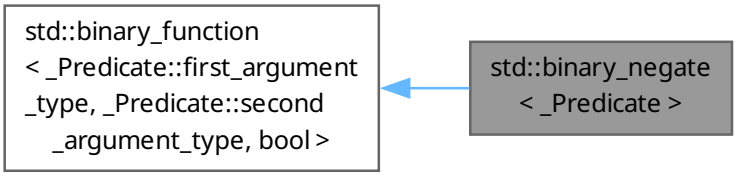
5.252.1 Detailed Description

Binary-heap (array-based).
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

5.253 std::binary_negate< _Predicate > Class Template Reference

`#include <stl_function.h>`
Inheritance diagram for std::binary_negate< _Predicate >:



Public Types

- typedef _Predicate::first_argument_type first_argument_type

- typedef bool result_type
- typedef _Predicate::second_argument_type second_argument_type

Public Member Functions

- constexpr **binary_negate** (const _Predicate &__x)
- constexpr bool **operator()** (const typename _Predicate::first_argument_type &__x, const typename _Predicate::second_argument_type &__y) const

Protected Attributes

- _Predicate **_M_pred**

5.253.1 Detailed Description

template<typename _Predicate>
class std::binary_negate< _Predicate >

One of the negation functors.

5.253.2 Member Typedef Documentation

first_argument_type

```
typedef _Predicate::first_argument_type std::binary_function< _Predicate::first_argument_type, _Predicate::second_argument_type, bool >::first_argument_type [inherited]
```

`first_argument_type` is the type of the first argument

result_type

```
typedef bool std::binary_function< _Predicate::first_argument_type, _Predicate::second_argument_type, bool >::result_type [inherited]
```

`result_type` is the return type

second_argument_type

```
typedef _Predicate::second_argument_type std::binary_function< _Predicate::first_argument_type, _Predicate::second_argument_type, bool >::second_argument_type [inherited]
```

`second_argument_type` is the type of the second argument

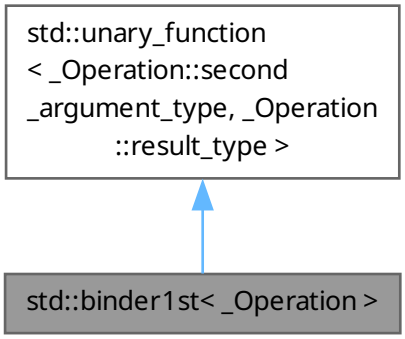
The documentation for this class was generated from the following file:

- stl_function.h

5.254 std::binder1st< _Operation > Class Template Reference

```
#include <binders.h>
```

Inheritance diagram for std::binder1st< _Operation >:



Public Types

- typedef _Operation::second_argument_type argument_type
- typedef _Operation::result_type result_type

Public Member Functions

- **binder1st** (const _Operation &__x, const typename _Operation::first_argument_type &__y)
- _Operation::result_type **operator()** (const typename _Operation::second_argument_type &__x) const
- _Operation::result_type **operator()** (typename _Operation::second_argument_type &__x) const

Protected Attributes

- _Operation **op**
- _Operation::first_argument_type **value**

5.254.1 Detailed Description

template<typename _Operation>
class std::binder1st< _Operation >

One of the binder functors.

5.254.2 Member Typedef Documentation

argument_type

```
typedef _Operation::second_argument_type std::unary_function< _Operation::second_argument_type, ↩
_Operation::result_type >::argument_type  [inherited]
```

`argument_type` is the type of the argument

result_type

```
typedef _Operation::result_type std::unary_function< _Operation::second_argument_type, _Operation↩
::result_type >::result_type [inherited]
```

result_type is the return type

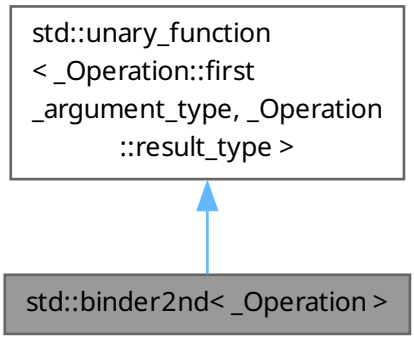
The documentation for this class was generated from the following file:

- backward/binders.h

5.255 std::binder2nd< _Operation > Class Template Reference

```
#include <binders.h>
```

Inheritance diagram for std::binder2nd< _Operation >:



Public Types

- typedef _Operation::first_argument_type argument_type
- typedef _Operation::result_type result_type

Public Member Functions

- **binder2nd** (const _Operation &__x, const typename _Operation::second_argument_type &__y)
- _Operation::result_type **operator()** (const typename _Operation::first_argument_type &__x) const
- _Operation::result_type **operator()** (typename _Operation::first_argument_type &__x) const

Protected Attributes

- _Operation **op**
- _Operation::second_argument_type **value**

5.255.1 Detailed Description

template<typename _Operation>
class std::binder2nd< _Operation >

One of the binder functors.

5.255.2 Member Typedef Documentation

argument_type

```
typedef _Operation::first_argument_type std::unary_function< _Operation::first_argument_type, _↩
Operation::result_type >::argument_type  [inherited]
```

`argument_type` is the type of the argument

result_type

```
typedef _Operation::result_type std::unary_function< _Operation::first_argument_type, _Operation↩
::result_type >::result_type  [inherited]
```

`result_type` is the return type

The documentation for this class was generated from the following file:

- backward/binders.h

5.256 std::binomial_distribution< _IntType > Class Template Reference

`#include <random>`

Classes

- struct param_type

Public Types

- typedef _IntType result_type

Public Member Functions

- **binomial_distribution** (_IntType __t, double __p=0.5)
- **binomial_distribution** (const param_type &__p)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- double p () const
- param_type param () const
- void param (const param_type &__param)
- void reset ()
- _IntType t () const

Friends

- template<typename _IntType1, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::binomial_distribution< _IntType1 > &__x)
- bool operator== (const binomial_distribution &__d1, const binomial_distribution &__d2)
- template<typename _IntType1, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::binomial_distribution< _IntType1 > &__x)

5.256.1 Detailed Description

template<typename _IntType = int>
class std::binomial_distribution< _IntType >

A discrete binomial random number distribution.
The formula for the binomial probability density function is $p(i|t,p) = \binom{t}{i} p^i (1-p)^{t-i}$ where t and p are the parameters of the distribution.

Since

C++11

5.256.2 Member Typedef Documentation

result_type

```
template<typename _IntType = int>
typedef _IntType std::binomial_distribution< _IntType >::result_type
```

The type of the range of the distribution.

5.256.3 Member Function Documentation

max()

```
template<typename _IntType = int>
result_type std::binomial_distribution< _IntType >::max () const  [inline]
```

Returns the least upper bound value of the distribution.

min()

```
template<typename _IntType = int>
result_type std::binomial_distribution< _IntType >::min () const  [inline]
```

Returns the greatest lower bound value of the distribution.

operator()() [1/2]

```
template<typename _IntType = int>
template<typename _UniformRandomNumberGenerator>
result_type std::binomial_distribution< _IntType >::operator() (
            _UniformRandomNumberGenerator & __urng)  [inline]
```

Generating functions.
Referenced by operator()().

operator()() [2/2]

```
template<typename _IntType>
template<typename _UniformRandomNumberGenerator>
```

```
binomial_distribution< _IntType >::result_type std::binomial_distribution< _IntType >::operator()
(
            _UniformRandomNumberGenerator & __urng,
            const param_type & __param)
```

A rejection algorithm when t * p >= 8 and a simple waiting time method - the second in the referenced book - otherwise.
NB: The former is available only if _GLIBCXX_USE_C99_MATH_FUNCS is defined.
Reference: Devroye, L. Non-Uniform Random Variates Generation. Springer-Verlag, New York, 1986, Ch. X, Sect. 4 (+ Errata!).
References std::abs(), std::numeric_limits< _Tp >::epsilon(), std::log(), and std::numeric_limits< _Tp >::max().

p()

```
template<typename _IntType = int>
double std::binomial_distribution< _IntType >::p () const  [inline]
```

Returns the distribution p parameter.

param() [1/2]

```
template<typename _IntType = int>
param_type std::binomial_distribution< _IntType >::param () const  [inline]
```

Returns the parameter set of the distribution.

param() [2/2]

```
template<typename _IntType = int>
void std::binomial_distribution< _IntType >::param (
            const param_type & __param)  [inline]
```

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

reset()

```
template<typename _IntType = int>
void std::binomial_distribution< _IntType >::reset ()  [inline]
```

Resets the distribution state.

t()

```
template<typename _IntType = int>
_IntType std::binomial_distribution< _IntType >::t () const  [inline]
```

Returns the distribution t parameter.

5.256.4 Friends And Related Symbol Documentation

operator<<

```
template<typename _IntType = int>
template<typename _IntType1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::binomial_distribution< _IntType1 > & __x)  [friend]
```

Inserts a binomial_distribution random number distribution __x into the output stream __os.

Parameters

__os	An output stream.
__x	A binomial_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

operator==

```
template<typename _IntType = int>
bool operator== (
             const binomial_distribution< _IntType > & __d1,
             const binomial_distribution< _IntType > & __d2)  [friend]
```

Return true if two binomial distributions have the same parameters and the sequences that would be generated are equal.

operator>>

```
template<typename _IntType = int>
template<typename _IntType1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
             std::basic_istream< _CharT, _Traits > & __is,
             std::binomial_distribution< _IntType1 > & __x)  [friend]
```

Extracts a binomial_distribution random number distribution __x from the input stream __is.

Parameters

_↩ _is	An input stream.
_↩ _x	A binomial_distribution random number generator engine.

Returns

The input stream with __x extracted or in an error state.

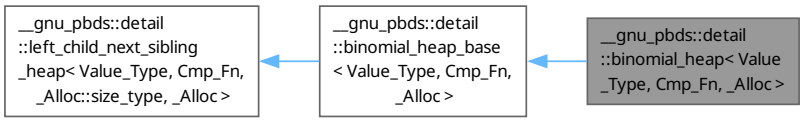
The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

5.257 __gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc > Class Template Reference

```
#include <binomial_heap_.hpp>
```

Inheritance diagram for __gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc >:



Public Types

- typedef base_type::allocator_type **allocator_type**
- typedef base_type::cmp_fn **cmp_fn**
- typedef base_type::const_iterator **const_iterator**
- typedef base_type::const_pointer **const_pointer**
- typedef base_type::const_reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef base_type::iterator **iterator**
- typedef base_type::point_const_iterator **point_const_iterator**
- typedef base_type::point_iterator **point_iterator**
- typedef base_type::pointer **pointer**
- typedef base_type::reference **reference**
- typedef _Alloc::size_type **size_type**
- typedef Value_Type **value_type**

Public Member Functions

- **binomial_heap** (const binomial_heap &)
- **binomial_heap** (const Cmp_Fn &)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- void **erase** (point_iterator)
- template<typename Pred>
  size_type **erase_if** (Pred)
- Cmp_Fn & **get_cmp_fn** ()
- const Cmp_Fn & **get_cmp_fn** () const
- void **join** (binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > &)
- size_type **max_size** () const
- void **modify** (point_iterator, const_reference)
- void **pop** ()
- point_iterator **push** (const_reference)
- size_type **size** () const
- template<typename Pred>
  void **split** (Pred, binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > &)
- void **swap** (left_child_next_sibling_heap< Value_Type, Cmp_Fn, _Alloc::size_type, _Alloc > &)
- const_reference **top** () const

Protected Types

- typedef base_type::node **node**
- typedef alloc_traits::allocator_type **node_allocator**
- typedef _Alloc::size_type **node_metadata**
- typedef std::pair< node_pointer, node_pointer > **node_pointer_pair**

Protected Member Functions

- void **actual_erase_node** (node_pointer)
- void **bubble_to_top** (node_pointer)
- void **clear_imp** (node_pointer)
- template<typename It>
  void **copy_from_range** (It, It)
- void **find_max** ()
- node_pointer **get_new_node_for_insert** (const_reference)
- node_pointer **prune** (Pred)
- void **swap** (binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > &)
- void **swap_with_parent** (node_pointer, node_pointer)
- void **to_linked_list** ()
- void **value_swap** (left_child_next_sibling_heap &)

Static Protected Member Functions

- static void **make_child_of** (node_pointer, node_pointer)
- static node_pointer **parent** (node_pointer)

Protected Attributes

- node_pointer **m_p_max**
- node_pointer **m_p_root**
- size_type **m_size**

5.257.1 Detailed Description

template<typename Value_Type, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc >

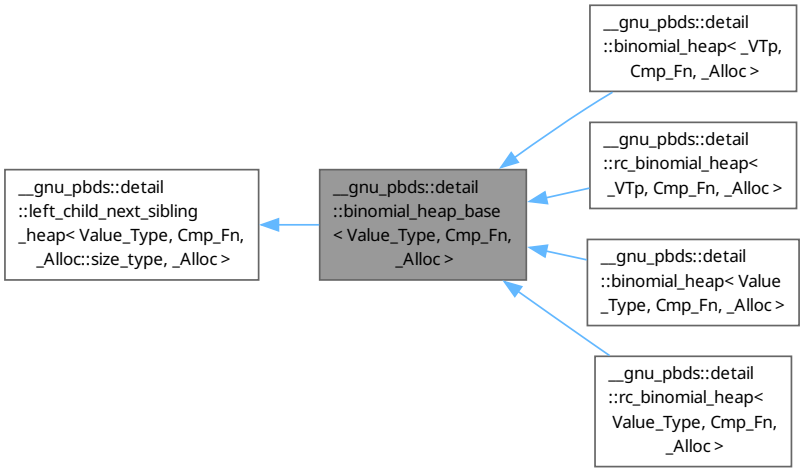
Binomial heap.
The documentation for this class was generated from the following file:

- binomial_heap_.hpp

5.258 __gnu_pbds::detail::binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > Class Template Reference

`#include <binomial_heap_base_.hpp>`

Inheritance diagram for __gnu_pbds::detail::binomial_heap_base< Value_Type, Cmp_Fn, _Alloc >:



Public Types

- typedef _Alloc **allocator_type**
- typedef Cmp_Fn **cmp_fn**
- typedef base_type::const_iterator **const_iterator**
- typedef __rebind_v::const_pointer **const_pointer**
- typedef __rebind_v::const_reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef base_type::iterator **iterator**
- typedef base_type::point_const_iterator **point_const_iterator**
- typedef base_type::point_iterator **point_iterator**
- typedef __rebind_v::pointer **pointer**
- typedef __rebind_v::reference **reference**
- typedef _Alloc::size_type **size_type**
- typedef Value_Type **value_type**

Public Member Functions

- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- void **erase** (point_iterator)
- template<typename Pred>
  size_type **erase_if** (Pred)
- Cmp_Fn & **get_cmp_fn** ()
- const Cmp_Fn & **get_cmp_fn** () const
- void **join** (binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > &)

- size_type **max_size** () const
- void **modify** (point_iterator, const_reference)
- void **pop** ()
- point_iterator **push** (const_reference)
- size_type **size** () const
- template<typename Pred>
  void **split** (Pred, binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > &)
- void **swap** (left_child_next_sibling_heap< Value_Type, Cmp_Fn, _Alloc::size_type, _Alloc > &)
- const_reference **top** () const

Protected Types

- typedef base_type::node **node**
- typedef alloc_traits::allocator_type **node_allocator**
- typedef base_type::node_const_pointer **node_const_pointer**
- typedef _Alloc::size_type **node_metadata**
- typedef base_type::node_pointer **node_pointer**
- typedef std::pair< node_pointer, node_pointer > **node_pointer_pair**

Protected Member Functions

- **binomial_heap_base** (const binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > &)
- **binomial_heap_base** (const Cmp_Fn &)
- void **actual_erase_node** (node_pointer)
- void **bubble_to_top** (node_pointer)
- void **clear_imp** (node_pointer)
- template<typename It>
  void **copy_from_range** (It, It)
- void **find_max** ()
- node_pointer **get_new_node_for_insert** (const_reference)
- node_pointer **prune** (Pred)
- void **swap** (binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > &)
- void **swap_with_parent** (node_pointer, node_pointer)
- void **to_linked_list** ()
- void **value_swap** (left_child_next_sibling_heap &)

Static Protected Member Functions

- static void **make_child_of** (node_pointer, node_pointer)
- static node_pointer **parent** (node_pointer)

Protected Attributes

- node_pointer **m_p_max**
- node_pointer **m_p_root**
- size_type **m_size**

5.258.1 Detailed Description

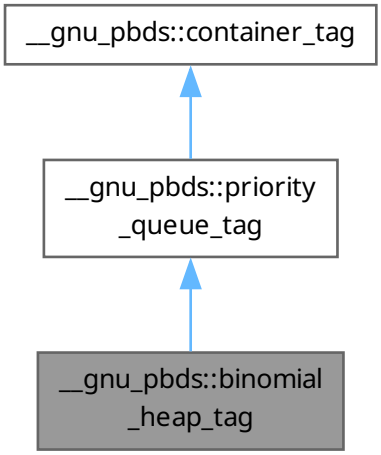
template<typename Value_Type, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::detail::binomial_heap_base< Value_Type, Cmp_Fn, _Alloc >

Base class for binomial heap.
The documentation for this class was generated from the following file:

- binomial_heap_base_.hpp

5.259 __gnu_pbds::binomial_heap_tag Struct Reference

`#include <tag_and_trait.hpp>`
Inheritance diagram for __gnu_pbds::binomial_heap_tag:



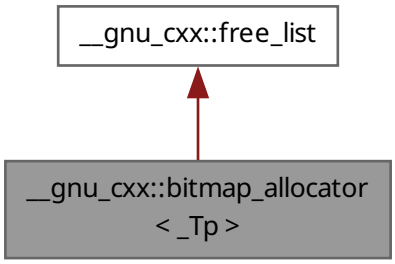
5.259.1 Detailed Description

Binomial-heap.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

5.260 __gnu_cxx::bitmap_allocator< _Tp > Class Template Reference

`#include <bitmap_allocator.h>`
Inheritance diagram for __gnu_cxx::bitmap_allocator< _Tp >:



Public Types

- typedef free_list::__mutex_type **__mutex_type**
- typedef const _Tp * **const_pointer**
- typedef const _Tp & **const_reference**
- typedef std::ptrdiff_t **difference_type**
- typedef _Tp * **pointer**
- typedef std::true_type **propagate_on_container_move_assignment**
- typedef _Tp & **reference**
- typedef std::size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **bitmap_allocator** (const bitmap_allocator &) noexcept
- template<typename _Tp1>
  bitmap_allocator (const bitmap_allocator< _Tp1 > &) noexcept
- pointer _M_allocate_single_object ()
- void _M_deallocate_single_object (pointer __p) throw ()
- const_pointer **address** (const_reference __r) const noexcept
- pointer **address** (reference __r) const noexcept
- pointer **allocate** (size_type __n)
- pointer **allocate** (size_type __n, typename bitmap_allocator< void >::const_pointer)
- template<typename _Up, typename... _Args>
  void **construct** (_Up *__p, _Args &&... __args)
- void **deallocate** (pointer __p, size_type __n) throw ()
- template<typename _Up>
  void **destroy** (_Up *__p)
- size_type **max_size** () const noexcept

5.260.1 Detailed Description

template<typename _Tp>
class __gnu_cxx::bitmap_allocator< _Tp >

Bitmap Allocator, primary template.

5.260.2 Member Function Documentation

_M_allocate_single_object()

```
template<typename _Tp>
pointer __gnu_cxx::bitmap_allocator< _Tp >::_M_allocate_single_object ()  [inline]
```

Allocates memory for a single object of size sizeof(_Tp).

Exceptions

std::bad_alloc.	If memory cannot be allocated.

Complexity: Worst case complexity is O(N), but that is hardly ever hit. If and when this particular case is encountered, the next few cases are guaranteed to have a worst case complexity of O(1)! That's why this function performs very well on average. You can consider this function to have a complexity referred to commonly as: Amortized Constant time.

_M_deallocate_single_object()

```
template<typename _Tp>
void __gnu_cxx::bitmap_allocator< _Tp >::_M_deallocate_single_object (
            pointer __p) throw ( )   [inline]
```

Deallocates memory that belongs to a single object of size sizeof(_Tp).
Complexity: O(lg(N)), but the worst case is not hit often! This is because containers usually deallocate memory close to each other and this case is handled in O(1) time by the deallocate function.
The documentation for this class was generated from the following file:

- bitmap_allocator.h

5.261 std::__debug::bitset< _Nb > Class Template Reference

```
#include <bitset>
```

Public Types

- typedef _Base::reference **reference**

Public Member Functions

- constexpr **bitset** (const _Base &__x)
- template<typename _CharT, typename = _Require<is_trivially_copyable<_CharT>, is_standard_layout<_CharT>, is_trivially_default_↩ constructible<_CharT>, __not_<is_array<_CharT>>>>
  constexpr **bitset** (const _CharT *__str, typename std::basic_string< _CharT >::size_type __n=std::basic_string< _CharT >::npos, _CharT __zero=_CharT('0'), _CharT __one=_CharT('1'))
- template<class _CharT, class _Traits, class _Alloc>
  constexpr **bitset** (const std::basic_string< _CharT, _Traits, _Alloc > &__str, typename std::basic_string< _↩ CharT, _Traits, _Alloc >::size_type __pos, typename std::basic_string< _CharT, _Traits, _Alloc >::size_type __n, _CharT __zero, _CharT __one=_CharT('1'))
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr **bitset** (const std::basic_string< _CharT, _Traits, _Alloc > &__str, typename std::basic_string< _↩ CharT, _Traits, _Alloc >::size_type __pos=0, typename std::basic_string< _CharT, _Traits, _Alloc >::size_type __n=(std::basic_string< _CharT, _Traits, _Alloc >::npos))
- constexpr **bitset** (unsigned long long __val) noexcept
- constexpr const _Base & **_M_base** () const noexcept
- constexpr _Base & **_M_base** () noexcept
- constexpr bitset< _Nb > & **flip** () noexcept
- constexpr bitset< _Nb > & **flip** (size_t __pos)
- constexpr bitset< _Nb > & **operator&=** (const bitset< _Nb > &__rhs) noexcept
- constexpr bitset< _Nb > **operator**<< (size_t __pos) const noexcept
- constexpr bitset< _Nb > & **operator**<<**=** (size_t __pos) noexcept
- constexpr bool **operator==** (const bitset< _Nb > &__rhs) const noexcept
- constexpr bitset< _Nb > **operator**>> (size_t __pos) const noexcept
- constexpr bitset< _Nb > & **operator**>>**=** (size_t __pos) noexcept
- constexpr reference **operator[ ]** (size_t __pos)
- constexpr bool **operator[ ]** (size_t __pos) const
- constexpr bitset< _Nb > & **operator^=** (const bitset< _Nb > &__rhs) noexcept
- constexpr bitset< _Nb > & **operator|=** (const bitset< _Nb > &__rhs) noexcept
- constexpr bitset< _Nb > **operator~** () const noexcept
- constexpr bitset< _Nb > & **reset** () noexcept
- constexpr bitset< _Nb > & **reset** (size_t __pos)
- constexpr bitset< _Nb > & **set** () noexcept

- constexpr bitset< _Nb > & **set** (size_t __pos, bool __val=true)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr std::basic_string< _CharT, _Traits, _Alloc > **to_string** () const
- template<typename _CharT, typename _Traits>
  constexpr std::basic_string< _CharT, _Traits, std::allocator< _CharT > > **to_string** () const
- template<typename _CharT>
  constexpr std::basic_string< _CharT, std::char_traits< _CharT >, std::allocator< _CharT > > **to_string** () const
- constexpr std::basic_string< char, std::char_traits< char >, std::allocator< char > > **to_string** () const
- template<class _CharT, class _Traits, class _Alloc>
  constexpr std::basic_string< _CharT, _Traits, _Alloc > **to_string** (_CharT __zero, _CharT __one=_CharT('1')) const
- template<class _CharT, class _Traits>
  constexpr std::basic_string< _CharT, _Traits, std::allocator< _CharT > > **to_string** (_CharT __zero, _CharT __one=_CharT('1')) const
- template<class _CharT>
  constexpr std::basic_string< _CharT, std::char_traits< _CharT >, std::allocator< _CharT > > **to_string** (_CharT __zero, _CharT __one=_CharT('1')) const
- constexpr std::basic_string< char, std::char_traits< char >, std::allocator< char > > **to_string** (char __zero, char __one='1') const

5.261.1 Detailed Description

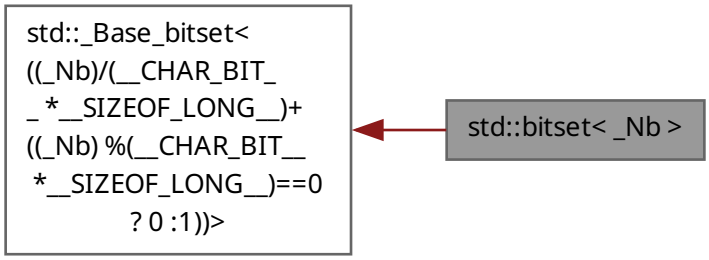
template<size_t _Nb>
class std::__debug::bitset< _Nb >

Class std::bitset with additional safety/checking/debug instrumentation.
The documentation for this class was generated from the following file:

- debug/bitset

5.262 std::bitset< _Nb > Class Template Reference

`#include <bitset>`
Inheritance diagram for std::bitset< _Nb >:



Classes

- class reference

Public Member Functions

- constexpr bitset () noexcept
- template<typename _CharT, typename = _Require<is_trivially_copyable<_CharT>, is_standard_layout<_CharT>, is_trivially_default_↩ constructible<_CharT>, __not_<is_array<_CharT>>>>
  constexpr bitset (const _CharT *__str, typename __bitset::__string< _CharT >::size_type __n=__bitset::__string< _CharT >::npos, _CharT __zero=_CharT('0'), _CharT __one=_CharT('1'))
- template<class _CharT, class _Traits, class _Alloc>
  constexpr bitset (const std::basic_string< _CharT, _Traits, _Alloc > &__s, size_t __position, size_t __n)
- template<class _CharT, class _Traits, class _Alloc>
  constexpr **bitset** (const std::basic_string< _CharT, _Traits, _Alloc > &__s, size_t __position, size_t __n, _CharT __zero, _CharT __one=_CharT('1'))
- template<class _CharT, class _Traits, class _Alloc>
  constexpr bitset (const std::basic_string< _CharT, _Traits, _Alloc > &__s, size_t __position=0)
- constexpr bitset (unsigned long long __val) noexcept
- constexpr size_t _Find_first () const noexcept
- constexpr size_t _Find_next (size_t __prev) const noexcept
- constexpr bool all () const noexcept
- constexpr bool any () const noexcept
- constexpr size_t count () const noexcept
- constexpr bitset< _Nb > & flip () noexcept
- constexpr bitset< _Nb > & flip (size_t __position)
- constexpr bool none () const noexcept
- constexpr bitset< _Nb > operator~ () const noexcept
- constexpr bitset< _Nb > & reset () noexcept
- constexpr bitset< _Nb > & reset (size_t __position)
- constexpr bitset< _Nb > & set () noexcept
- constexpr bitset< _Nb > & set (size_t __position, bool __val=true)
- constexpr size_t size () const noexcept
- constexpr bool test (size_t __position) const
- template<class _CharT, class _Traits, class _Alloc>
  constexpr std::basic_string< _CharT, _Traits, _Alloc > to_string () const
- template<class _CharT, class _Traits>
  constexpr std::basic_string< _CharT, _Traits, std::allocator< _CharT > > **to_string** () const
- template<class _CharT>
  constexpr std::basic_string< _CharT, std::char_traits< _CharT >, std::allocator< _CharT > > **to_string** () const
- constexpr std::basic_string< char, std::char_traits< char >, std::allocator< char > > **to_string** () const
- template<class _CharT, class _Traits, class _Alloc>
  constexpr std::basic_string< _CharT, _Traits, _Alloc > **to_string** (_CharT __zero, _CharT __one=_CharT('1')) const
- template<class _CharT, class _Traits>
  constexpr std::basic_string< _CharT, _Traits, std::allocator< _CharT > > **to_string** (_CharT __zero, _CharT __one=_CharT('1')) const
- template<class _CharT>
  constexpr std::basic_string< _CharT, std::char_traits< _CharT >, std::allocator< _CharT > > **to_string** (_↩ CharT __zero, _CharT __one=_CharT('1')) const
- constexpr std::basic_string< char, std::char_traits< char >, std::allocator< char > > **to_string** (char __zero, char __one='1') const
- constexpr unsigned long long **to_ullong** () const
- constexpr unsigned long to_ulong () const

- constexpr bitset< _Nb > & operator&= (const bitset< _Nb > &__rhs) noexcept

- constexpr bitset< _Nb > & operator|= (const bitset< _Nb > &__rhs) noexcept
- constexpr bitset< _Nb > & operator^= (const bitset< _Nb > &__rhs) noexcept

- constexpr bitset< _Nb > & operator<<= (size_t __position) noexcept
- constexpr bitset< _Nb > & operator>>= (size_t __position) noexcept

- constexpr bitset< _Nb > & _Unchecked_set (size_t __pos) noexcept
- constexpr bitset< _Nb > & _Unchecked_set (size_t __pos, int __val) noexcept
- constexpr bitset< _Nb > & _Unchecked_reset (size_t __pos) noexcept
- constexpr bitset< _Nb > & _Unchecked_flip (size_t __pos) noexcept
- constexpr bool _Unchecked_test (size_t __pos) const noexcept

- constexpr reference operator[ ] (size_t __position)
- constexpr bool operator[ ] (size_t __position) const

- constexpr bool operator== (const bitset< _Nb > &__rhs) const noexcept

- constexpr bitset< _Nb > operator<< (size_t __position) const noexcept
- constexpr bitset< _Nb > operator>> (size_t __position) const noexcept

Friends

- template<class _CharT, class _Traits, size_t _Nb2>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &, const bitset< _Nb2 > &)
- template<class _CharT, class _Traits, size_t _Nb2>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &, bitset< _Nb2 > &)
- class **reference**
- struct **std::hash**< **bitset** >

5.262.1 Detailed Description

template<**size_t _Nb**>
class std::bitset< **_Nb** >

The bitset class represents a *fixed-size* sequence of bits.
(Note that bitset does *not* meet the formal requirements of a `container`. Mainly, it lacks iterators.)
The template argument, `Nb`, may be any non-negative number, specifying the number of bits (e.g., "0", "12", "1024*1024").
In the general unoptimized case, storage is allocated in word-sized blocks. Let B be the number of bits in a word, then (Nb+(B-1))/B words will be used for storage. B - NbB bits are unused. (They are the high-order bits in the highest word.) It is a class invariant that those unused bits are always zero.
If you think of bitset as *a simple array of bits*, be aware that your mental picture is reversed: a bitset behaves the same way as bits in integers do, with the bit at index 0 in the *least significant / right-hand* position, and the bit at index Nb-1 in the *most significant / left-hand* position. Thus, unlike other containers, a bitset's index *counts from right to left*, to put it very loosely.

This behavior is preserved when translating to and from strings. For example, the first line of the following program probably prints *b('a') is 0001100001* on a modern ASCII system.

```
#include <bitset>
#include <iostream>
#include <sstream>

using namespace std;

int main()
{
    long         a = 'a';
    bitset<10>   b(a);

    cout « "b('a') is " « b « endl;

    ostringstream s;
    s « b;
    string  str = s.str();
    cout « "index 3 in the string is " « str[3] « " but\n"
         « "index 3 in the bitset is " « b[3] « endl;
}
```

Also see: https://gcc.gnu.org/onlinedocs/libstdc++/manual/ext_containers.html for a description of extensions.

Most of the actual code isn't contained in bitset<> itself, but in the base class _Base_bitset. The base class works with whole words, not with individual bits. This allows us to specialize _Base_bitset for the important special case where the bitset is only a single word.

Extra confusion can result due to the fact that the storage for _Base_bitset *is* a regular array, and is indexed as such. This is carefully encapsulated.

5.262.2 Constructor & Destructor Documentation

bitset() [1/5]

```
template<size_t _Nb>
std::bitset< _Nb >::bitset ()  [inline], [constexpr], [noexcept]
```

All bits set to zero.

bitset() [2/5]

```
template<size_t _Nb>
std::bitset< _Nb >::bitset (
            unsigned long long __val)  [inline], [constexpr], [noexcept]
```

Initial bits bitwise-copied from a single word (others set to zero).

bitset() [3/5]

```
template<size_t _Nb>
template<class _CharT, class _Traits, class _Alloc>
std::bitset< _Nb >::bitset (
            const std::basic_string< _CharT, _Traits, _Alloc > & __s,
            size_t __position = 0)  [inline], [explicit], [constexpr]
```

Use a subset of a string.

Parameters

__s	A string of 0 and 1 characters.
__position	Index of the first character in __s to use; defaults to zero.

Exceptions

std::out_of_range	If __position > __s.size().
std::invalid_argument	If a character appears in the string which is neither 0 nor 1.

bitset() [4/5]

```
template<size_t _Nb>
template<class _CharT, class _Traits, class _Alloc>
std::bitset< _Nb >::bitset (
            const std::basic_string< _CharT, _Traits, _Alloc > & __s,
            size_t __position,
            size_t __n)  [inline], [constexpr]
```

Use a subset of a string.

Parameters

__s	A string of 0 and 1 characters.
__position	Index of the first character in __s to use.
__n	The number of characters to copy.

Exceptions

std::out_of_range	If __position > __s.size().
std::invalid_argument	If a character appears in the string which is neither 0 nor 1.

bitset() [5/5]

```
template<size_t _Nb>
template<typename _CharT, typename = _Require<is_trivially_copyable<_CharT>, is_standard_↩
layout<_CharT>, is_trivially_default_constructible<_CharT>, __not_<is_array<_CharT>>>>
std::bitset< _Nb >::bitset (
            const _CharT * __str,
            typename __bitset< _Nb >::__string< _CharT >::size_type __n = __bitset< _Nb >::__↩
string<_CharT>::npos,
            _CharT __zero = _CharT('0'),
            _CharT __one = _CharT('1'))  [inline], [explicit], [constexpr]
```

Construct from a character array.

Parameters

__str	An array of characters __zero and __one.
__n	The number of characters to use.
__zero	The character corresponding to the value 0.
__one	The character corresponding to the value 1.

Exceptions

std::invalid_argument	If a character appears in the string which is neither __zero nor __one.

5.262.3 Member Function Documentation

all()

```
template<size_t _Nb>
bool std::bitset< _Nb >::all () const  [inline], [constexpr], [noexcept]
```

Tests whether all the bits are on.

Returns

True if all the bits are set.

any()

```
template<size_t _Nb>
bool std::bitset< _Nb >::any () const  [inline], [constexpr], [noexcept]
```

Tests whether any of the bits are on.

Returns

True if at least one bit is set.

count()

```
template<size_t _Nb>
size_t std::bitset< _Nb >::count () const  [inline], [constexpr], [noexcept]
```

Returns the number of bits which are set.

flip() [1/2]

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::flip ()  [inline], [constexpr], [noexcept]
```

Toggles every bit to its opposite value.

flip() [2/2]

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::flip (
            size_t __position)  [inline], [constexpr]
```

Toggles a given bit to its opposite value.

Parameters

__position	The index of the bit.

Exceptions

std::out_of_range	If *pos* is bigger the size of the set.

none()

```
template<size_t _Nb>
bool std::bitset< _Nb >::none () const  [inline], [constexpr], [noexcept]
```

Tests whether any of the bits are on.

Returns

True if none of the bits are set.

operator&=()

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::operator&= (
            const bitset< _Nb > & __rhs)  [inline], [constexpr], [noexcept]
```

Operations on bitsets.

Parameters

__rhs	A same-sized bitset.

These should be self-explanatory.

operator<<()

```
template<size_t _Nb>
bitset< _Nb > std::bitset< _Nb >::operator<< (
            size_t __position) const  [inline], [constexpr], [noexcept]
```

Self-explanatory.

operator<<=()

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::operator<<= (
            size_t __position)  [inline], [constexpr], [noexcept]
```

Operations on bitsets.

Parameters

__position	The number of places to shift.

These should be self-explanatory.

operator==()

```
template<size_t _Nb>
bool std::bitset< _Nb >::operator== (
            const bitset< _Nb > & __rhs) const  [inline], [constexpr], [noexcept]
```

These comparisons for equality/inequality are, well, *bitwise*.

operator>>()

```
template<size_t _Nb>
bitset< _Nb > std::bitset< _Nb >::operator>> (
            size_t __position) const  [inline], [constexpr], [noexcept]
```

Self-explanatory.

operator>>=()

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::operator>>= (
            size_t __position)  [inline], [constexpr], [noexcept]
```

Operations on bitsets.

Parameters

__position	The number of places to shift.

These should be self-explanatory.

operator[]() [1/2]

```
template<size_t _Nb>
reference std::bitset< _Nb >::operator[] (
            size_t __position)  [inline], [constexpr]
```

Array-indexing support.

Parameters

__position	Index into the bitset.

Returns

A bool for a *const bitset*. For non-const bitsets, an instance of the reference proxy class.

Note

These operators do no range checking and throw no exceptions, as required by DR 11 to the standard.

_GLIBCXX_RESOLVE_LIB_DEFECTS Note that this implementation already resolves DR 11 (items 1 and 2), but does not do the range-checking required by that DR's resolution. -pme The DR has since been changed: range-checking is a precondition (users' responsibility), and these functions must not throw. -pme

operator[]() [2/2]

```
template<size_t _Nb>
bool std::bitset< _Nb >::operator[] (
            size_t __position) const  [inline], [constexpr]
```

Array-indexing support.

Parameters

__position	Index into the bitset.

Returns

A bool for a *const bitset*. For non-const bitsets, an instance of the reference proxy class.

Note

These operators do no range checking and throw no exceptions, as required by DR 11 to the standard.

_GLIBCXX_RESOLVE_LIB_DEFECTS Note that this implementation already resolves DR 11 (items 1 and 2), but does not do the range-checking required by that DR's resolution. -pme The DR has since been changed: range-checking is a precondition (users' responsibility), and these functions must not throw. -pme

operator^=()

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::operator^= (
            const bitset< _Nb > & __rhs)  [inline], [constexpr], [noexcept]
```

Operations on bitsets.

Parameters

__rhs	A same-sized bitset.

These should be self-explanatory.

operator"" |=()

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::operator|= (
            const bitset< _Nb > & __rhs)  [inline], [constexpr], [noexcept]
```

Operations on bitsets.

Parameters

__rhs	A same-sized bitset.

These should be self-explanatory.

operator~()

```
template<size_t _Nb>
bitset< _Nb > std::bitset< _Nb >::operator~ () const  [inline], [constexpr], [noexcept]
```

See the no-argument flip().

reset() [1/2]

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::reset ()  [inline], [constexpr], [noexcept]
```

Sets every bit to false.

reset() [2/2]

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::reset (
            size_t __position)  [inline], [constexpr]
```

Sets a given bit to false.

Parameters

__position	The index of the bit.

Exceptions

std::out_of_range	If *pos* is bigger the size of the set.

Same as writing `set(pos,false)`.

set() [1/2]

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::set ()  [inline], [constexpr], [noexcept]
```

Sets every bit to true.

set() [2/2]

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::set (
            size_t __position,
            bool __val = true)  [inline], [constexpr]
```

Sets a given bit to a particular value.

Parameters

__position	The index of the bit.
__val	Either true or false, defaults to true.

Exceptions

std::out_of_range	If *pos* is bigger the size of the set.

size()

```
template<size_t _Nb>
size_t std::bitset< _Nb >::size () const  [inline], [constexpr], [noexcept]
```

Returns the total number of bits.

test()

```
template<size_t _Nb>
bool std::bitset< _Nb >::test (
            size_t __position) const  [inline], [constexpr]
```

Tests the value of a bit.

Parameters

__position	The index of a bit.

Returns

The value at *pos*.

Exceptions

std::out_of_range	If *pos* is bigger the size of the set.

to_string()

```
template<size_t _Nb>
template<class _CharT, class _Traits, class _Alloc>
std::basic_string< _CharT, _Traits, _Alloc > std::bitset< _Nb >::to_string () const  [inline],
[constexpr]
```

Returns a character interpretation of the bitset.

Returns

The string equivalent of the bits.

Note the ordering of the bits: decreasing character positions correspond to increasing bit positions (see the main class notes for an example).

to_ulong()

```
template<size_t _Nb>
unsigned long std::bitset< _Nb >::to_ulong () const  [inline], [constexpr]
```

Returns a numerical interpretation of the bitset.

Returns

The integral equivalent of the bits.

Exceptions

std::overflow_error	If there are too many bits to be represented in an `unsigned long`.

The documentation for this class was generated from the following file:

- bitset

5.263 std::tr2::bool_set Class Reference

```
#include <bool_set>
```

Public Member Functions

- constexpr bool_set ()
- constexpr bool_set (bool __t)
- bool **contains** (bool_set __b) const
- bool equals (bool_set __b) const
- bool is_emptyset () const
- bool is_indeterminate () const
- bool is_singleton () const
- operator bool () const

Static Public Member Functions

- static bool_set **emptyset** ()
- static bool_set **indeterminate** ()

Friends

- bool_set **operator!** (bool_set __b)
- bool_set **operator&** (bool_set __s, bool_set __t)
- template<typename CharT, typename Traits>
  std::basic_ostream< CharT, Traits > & **operator**<< (std::basic_ostream< CharT, Traits > &__out, bool_set __b)
- bool_set **operator==** (bool_set __s, bool_set __t)
- template<typename CharT, typename Traits>
  std::basic_istream< CharT, Traits > & **operator**>> (std::basic_istream< CharT, Traits > &__in, bool_set &__b)
- bool_set **operator^** (bool_set __s, bool_set __t)
- bool_set **operator|** (bool_set __s, bool_set __t)

5.263.1 Detailed Description

bool_set
See N2136, Bool_set: multi-valued logic by Hervé Brönnimann, Guillaume Melquiond, Sylvain Pion.
The implicit conversion to bool is slippery! I may use the new explicit conversion. This has been specialized in the language so that in contexts requiring a bool the conversion happens implicitly. Thus most objections should be eliminated.

5.263.2 Constructor & Destructor Documentation

bool_set() [1/2]

```
std::tr2::bool_set::bool_set ()  [inline], [constexpr]
```

Default constructor.

bool_set() [2/2]

```
std::tr2::bool_set::bool_set (
            bool __t)  [inline], [constexpr]
```

Constructor from bool.

5.263.3 Member Function Documentation

equals()

```
bool std::tr2::bool_set::equals (
            bool_set __b) const  [inline]
```

Return true if states are equal.

is_emptyset()

```
bool std::tr2::bool_set::is_emptyset () const  [inline]
```

Return true if this is empty.

is_indeterminate()

```
bool std::tr2::bool_set::is_indeterminate () const  [inline]
```

Return true if this is indeterminate.

is_singleton()

```
bool std::tr2::bool_set::is_singleton () const  [inline]
```

Return true if this is false or true (normal boolean).

operator bool()

```
std::tr2::bool_set::operator bool () const  [inline]
```

Conversion to bool.

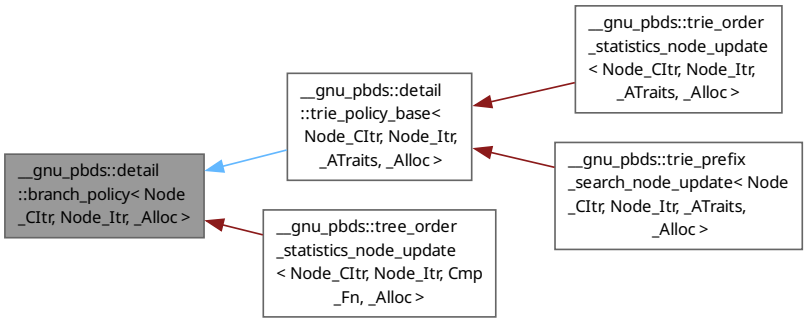
The documentation for this class was generated from the following files:

- bool_set
- bool_set.tcc

5.264 __gnu_pbds::detail::branch_policy< Node_CItr, Node_Itr, _Alloc > Struct Template Reference

```
#include <branch_policy.hpp>
```

Inheritance diagram for __gnu_pbds::detail::branch_policy< Node_CItr, Node_Itr, _Alloc >:



Protected Types

- typedef rebind_v::const_pointer **const_pointer**
- typedef rebind_v::const_reference **const_reference**
- typedef Node_Itr::value_type **it_type**
- typedef rebind_k::const_reference **key_const_reference**
- typedef value_type::first_type **key_type**
- typedef remove_const< key_type >::type **rckey_type**
- typedef remove_const< value_type >::type **rcvalue_type**
- typedef rebind_traits< _Alloc, rckey_type > **rebind_k**
- typedef rebind_traits< _Alloc, rcvalue_type > **rebind_v**
- typedef rebind_v::reference **reference**
- typedef std::iterator_traits< it_type >::value_type **value_type**

Protected Member Functions

- virtual it_type **end** ()=0
- it_type **end_iterator** () const

Static Protected Member Functions

- static key_const_reference **extract_key** (const_reference r_val)

5.264.1 Detailed Description

template<typename Node_CItr, typename Node_Itr, typename _Alloc>
struct __gnu_pbds::detail::branch_policy< Node_CItr, Node_Itr, _Alloc >

Primary template, base class for branch structure policies.
The documentation for this struct was generated from the following file:

- branch_policy.hpp

5.265 __gnu_pbds::detail::branch_policy< Node_CItr, Node_CItr, _Alloc > Struct Template Reference

`#include <branch_policy.hpp>`

Protected Types

- typedef rebind_v::const_pointer **const_pointer**
- typedef rebind_v::const_pointer **const_pointer**
- typedef rebind_v::const_reference **const_reference**
- typedef rebind_v::const_reference **const_reference**
- typedef Node_CItr::value_type **it_type**
- typedef Node_CItr::value_type **it_type**
- typedef rebind_k::const_reference **key_const_reference**
- typedef rebind_v::const_reference **key_const_reference**
- typedef value_type::first_type **key_type**
- typedef value_type **key_type**
- typedef remove_const< key_type >::type **rckey_type**
- typedef remove_const< value_type >::type **rcvalue_type**
- typedef remove_const< value_type >::type **rcvalue_type**
- typedef rebind_traits< _Alloc, rckey_type > **rebind_k**
- typedef rebind_traits< _Alloc, rcvalue_type > **rebind_v**
- typedef rebind_traits< _Alloc, rcvalue_type > **rebind_v**
- typedef rebind_v::reference **reference**
- typedef rebind_v::reference **reference**
- typedef std::iterator_traits< it_type >::value_type **value_type**
- typedef std::iterator_traits< it_type >::value_type **value_type**

Protected Member Functions

- virtual it_type **end** () const =0
- virtual it_type **end** ()=0
- it_type **end_iterator** () const
- it_type **end_iterator** () const

Static Protected Member Functions

- static key_const_reference **extract_key** (const_reference r_val)
- static key_const_reference **extract_key** (const_reference r_val)

5.265.1 Detailed Description

template<typename Node_Cltr, typename _Alloc>
struct __gnu_pbds::detail::branch_policy< Node_Cltr, Node_Cltr, _Alloc >

Specialization for const iterators.
The documentation for this struct was generated from the following file:

- branch_policy.hpp

5.266 std::cauchy_distribution< _RealType > Class Template Reference

`#include <random>`

Classes

- struct param_type

Public Types

- typedef _RealType result_type

Public Member Functions

- **cauchy_distribution** (_RealType __a, _RealType __b=1.0)
- **cauchy_distribution** (const param_type &__p)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- _RealType **a** () const
- _RealType **b** () const
- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- bool operator== (const cauchy_distribution &__d1, const cauchy_distribution &__d2)

5.266.1 Detailed Description

template<typename _RealType = double>
class std::cauchy_distribution< _RealType >

A cauchy_distribution random number distribution.
The formula for the normal probability mass function is $p(x|a,b) = (\pi b(1+(\frac{x-a}{b})^2))^{-1}$

Since

C++11

5.266.2 Member Typedef Documentation

result_type

```
template<typename _RealType = double>
typedef _RealType std::cauchy_distribution< _RealType >::result_type
```

The type of the range of the distribution.

5.266.3 Member Function Documentation

max()

```
template<typename _RealType = double>
result_type std::cauchy_distribution< _RealType >::max () const  [inline]
```

Returns the least upper bound value of the distribution.

min()

```
template<typename _RealType = double>
result_type std::cauchy_distribution< _RealType >::min () const  [inline]
```

Returns the greatest lower bound value of the distribution.

operator()()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::cauchy_distribution< _RealType >::operator() (
            _UniformRandomNumberGenerator & __urng)  [inline]
```

Generating functions.
Referenced by operator()().

param() [1/2]

```
template<typename _RealType = double>
param_type std::cauchy_distribution< _RealType >::param () const  [inline]
```

Returns the parameter set of the distribution.
Referenced by std::operator>>().

param() [2/2]

```
template<typename _RealType = double>
void std::cauchy_distribution< _RealType >::param (
            const param_type & __param)  [inline]
```

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

reset()

```
template<typename _RealType = double>
void std::cauchy_distribution< _RealType >::reset ()  [inline]
```

Resets the distribution state.

5.266.4 Friends And Related Symbol Documentation

operator==

```
template<typename _RealType = double>
bool operator== (
            const cauchy_distribution< _RealType > & __d1,
            const cauchy_distribution< _RealType > & __d2)  [friend]
```

Return true if two Cauchy distributions have the same parameters.
The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

5.267 __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type > Class Template Reference

```
#include <hash_policy.hpp>
```

Public Types

- enum { external_load_access }
- typedef Size_Type **size_type**

Public Member Functions

- cc_hash_max_collision_check_resize_trigger (float load=0.5)
- float get_load () const
- void set_load (float load)
- void **swap** (cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type > &other)

Protected Member Functions

- bool is_grow_needed (size_type size, size_type num_entries) const
- bool is_resize_needed () const
- void notify_cleared ()
- void notify_erase_search_collision ()
- void notify_erase_search_end ()
- void notify_erase_search_start ()
- void notify_erased (size_type num_entries)
- void notify_externally_resized (size_type new_size)
- void notify_find_search_collision ()
- void notify_find_search_end ()

- void notify_find_search_start ()
- void notify_insert_search_collision ()
- void notify_insert_search_end ()
- void notify_insert_search_start ()
- void notify_inserted (size_type num_entries)
- void notify_resized (size_type new_size)

5.267.1 Detailed Description

template<bool External_Load_Access = false, typename Size_Type = std::size_t>
class __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >

A resize trigger policy based on collision checks. It keeps the simulated load factor lower than some given load factor.

5.267.2 Member Enumeration Documentation

anonymous enum

```
template<bool External_Load_Access = false, typename Size_Type = std::size_t>
anonymous enum
```

Enumerator

external_load_access	Specifies whether the load factor can be accessed externally. The two options have different trade-offs in terms of flexibility, genericity, and encapsulation.

5.267.3 Constructor & Destructor Documentation

cc_hash_max_collision_check_resize_trigger()

```
template<bool External_Load_Access, typename Size_Type>
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::cc_↩
hash_max_collision_check_resize_trigger (
            float load = 0.5)
```

Default constructor, or constructor taking load, a __load factor which it will attempt to maintain.

5.267.4 Member Function Documentation

get_load()

```
template<bool External_Load_Access, typename Size_Type>
float __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↩
::get_load () const  [inline]
```

Returns the current load.

is_grow_needed()

```
template<bool External_Load_Access, typename Size_Type>
bool __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↩
::is_grow_needed (
            size_type size,
            size_type num_entries) const  [inline], [protected]
```

Queries whether a grow is needed. This method is called only if this object indicated is needed.

is_resize_needed()

```
template<bool External_Load_Access, typename Size_Type>
bool __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↩
::is_resize_needed () const  [inline], [protected]
```

Queries whether a resize is needed.

notify_cleared()

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↩
::notify_cleared ()  [protected]
```

Notifies the table was cleared.

References notify_resized().

notify_erase_search_collision()

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↩
::notify_erase_search_collision ()  [inline], [protected]
```

Notifies a search encountered a collision.

notify_erase_search_end()

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↩
::notify_erase_search_end ()  [inline], [protected]
```

Notifies a search ended.

notify_erase_search_start()

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↩
::notify_erase_search_start ()  [inline], [protected]
```

Notifies an erase search started.

notify_erased()

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↩
::notify_erased (
            size_type num_entries)  [inline], [protected]
```

Notifies an element was erased.

notify_externally_resized()

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↩
::notify_externally_resized (
            size_type new_size)  [protected]
```

Notifies the table was resized externally.

notify_find_search_collision()

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↩
::notify_find_search_collision () [inline], [protected]
```

Notifies a search encountered a collision.

notify_find_search_end()

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↩
::notify_find_search_end () [inline], [protected]
```

Notifies a search ended.

notify_find_search_start()

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↩
::notify_find_search_start () [inline], [protected]
```

Notifies a find search started.

notify_insert_search_collision()

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↩
::notify_insert_search_collision () [inline], [protected]
```

Notifies a search encountered a collision.

notify_insert_search_end()

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↩
::notify_insert_search_end () [inline], [protected]
```

Notifies a search ended.

notify_insert_search_start()

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↩
::notify_insert_search_start () [inline], [protected]
```

Notifies an insert search started.

notify_inserted()

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↩
::notify_inserted (
            size_type num_entries) [inline], [protected]
```

Notifies an element was inserted.

notify_resized()

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↩
::notify_resized (
            size_type new_size) [protected]
```

Notifies the table was resized as a result of this object's signifying that a resize is needed.
Referenced by notify_cleared().

set_load()

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >↩
::set_load (
            float load)
```

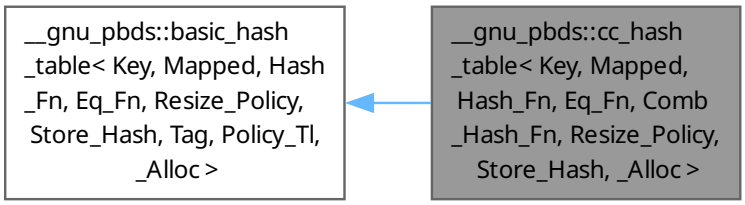
Sets the load; does not resize the container.
The documentation for this class was generated from the following file:

- hash_policy.hpp

5.268 __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc > Class Template Reference

```
#include <assoc_container.hpp>
```

Inheritance diagram for __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >:



Public Types

- typedef Comb_Hash_Fn **comb_hash_fn**
- typedef cc_hash_tag **container_category**
- typedef Eq_Fn **eq_fn**
- typedef Hash_Fn **hash_fn**
- typedef Resize_Policy **resize_policy**

Public Member Functions

- cc_hash_table ()
- **cc_hash_table** (const cc_hash_table &other)
- cc_hash_table (const hash_fn &h)
- cc_hash_table (const hash_fn &h, const eq_fn &e)
- cc_hash_table (const hash_fn &h, const eq_fn &e, const comb_hash_fn &ch)
- cc_hash_table (const hash_fn &h, const eq_fn &e, const comb_hash_fn &ch, const resize_policy &rp)
- template<typename It>
  cc_hash_table (It first, It last)

- template<typename It>
  cc_hash_table (It first, It last, const hash_fn &h)
- template<typename It>
  cc_hash_table (It first, It last, const hash_fn &h, const eq_fn &e)
- template<typename It>
  cc_hash_table (It first, It last, const hash_fn &h, const eq_fn &e, const comb_hash_fn &ch)
- template<typename It>
  cc_hash_table (It first, It last, const hash_fn &h, const eq_fn &e, const comb_hash_fn &ch, const resize_policy &rp)
- cc_hash_table & **operator=** (const cc_hash_table &other)
- void **swap** (cc_hash_table &other)

5.268.1 Detailed Description

template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
class __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >

A collision-chaining hash-based associative container.

Template Parameters

Key	Key type.
Mapped	Map type.
Hash_Fn	Hashing functor.
Eq_Fn	Equal functor.
Comb_Hash_Fn	Combining hash functor. If Hash_Fn is not null_type, then this is the ranged-hash functor; otherwise, this is the range-hashing functor. XXX(See Design::Hash-Based Containers::Hash Policies.)
Resize_Policy	Resizes hash.
Store_Hash	Indicates whether the hash value will be stored along with each key. If Hash_Fn is null_type, then the container will not compile if this value is true
_Alloc	Allocator type.

Base tag choices are: cc_hash_tag.
Base is basic_hash_table.

5.268.2 Constructor & Destructor Documentation

cc_hash_table() [1/10]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↩
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn =
detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_↩
policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std↩
::allocator<char>>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table ()  [inline]
```

Default constructor.

cc_hash_table() [2/10]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↩
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn =
detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_↩
policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std↩
::allocator<char>>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
            const hash_fn & h)  [inline]
```

Constructor taking some policy objects. r_hash_fn will be copied by the Hash_Fn object of the container object.

cc_hash_table() [3/10]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↩
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn =
detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_↩
policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std↩
::allocator<char>>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
            const hash_fn & h,
            const eq_fn & e)  [inline]
```

Constructor taking some policy objects. r_hash_fn will be copied by the hash_fn object of the container object, and r_eq_fn will be copied by the eq_fn object of the container object.

cc_hash_table() [4/10]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↩
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn =
detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_↩
policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std↩
::allocator<char>>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
            const hash_fn & h,
            const eq_fn & e,
            const comb_hash_fn & ch)  [inline]
```

Constructor taking some policy objects. r_hash_fn will be copied by the hash_fn object of the container object, r_eq_fn will be copied by the eq_fn object of the container object, and r_comb_hash_fn will be copied by the comb_hash_fn object of the container object.

cc_hash_table() [5/10]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↩
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn =
detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_↩
policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std↩
::allocator<char>>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
            const hash_fn & h,
            const eq_fn & e,
            const comb_hash_fn & ch,
            const resize_policy & rp)  [inline]
```

Constructor taking some policy objects. r_hash_fn will be copied by the hash_fn object of the container object, r_eq_fn will be copied by the eq_fn object of the container object, r_comb_hash_fn will be copied by the comb_hash_fn object of the container object, and r_resize_policy will be copied by the resize_policy object of the container object.

cc_hash_table() [6/10]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↩
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn =
detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_↩
policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std↩
::allocator<char>>
template<typename It>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
            It first,
            It last)  [inline]
```

Constructor taking __iterators to a range of value_types. The value_types between first_it and last_it will be inserted into the container object.

cc_hash_table() [7/10]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↩
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn =
detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_↩
policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std↩
::allocator<char>>
template<typename It>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
            It first,
            It last,
            const hash_fn & h)  [inline]
```

Constructor taking __iterators to a range of value_types and some policy objects. The value_types between first_it and last_it will be inserted into the container object.

cc_hash_table() [8/10]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↩
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn =
detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_↩
policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std↩
::allocator<char>>
template<typename It>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
            It first,
            It last,
            const hash_fn & h,
            const eq_fn & e)  [inline]
```

Constructor taking __iterators to a range of value_types and some policy objects The value_types between first_it and last_it will be inserted into the container object. r_hash_fn will be copied by the hash_fn object of the container object, and r_eq_fn will be copied by the eq_fn object of the container object.

cc_hash_table() [9/10]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↩
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn =
detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_↩
policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std↩
::allocator<char>>
template<typename It>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
             It first,
             It last,
             const hash_fn & h,
             const eq_fn & e,
             const comb_hash_fn & ch)  [inline]
```

Constructor taking __iterators to a range of value_types and some policy objects The value_types between first_↩ it and last_it will be inserted into the container object. r_hash_fn will be copied by the hash_fn object of the container object, r_eq_fn will be copied by the eq_fn object of the container object, and r_comb_hash_fn will be copied by the comb_hash_fn object of the container object.

cc_hash_table() [10/10]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↩
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn =
detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_↩
policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std↩
::allocator<char>>
template<typename It>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
             It first,
             It last,
             const hash_fn & h,
             const eq_fn & e,
             const comb_hash_fn & ch,
             const resize_policy & rp)  [inline]
```

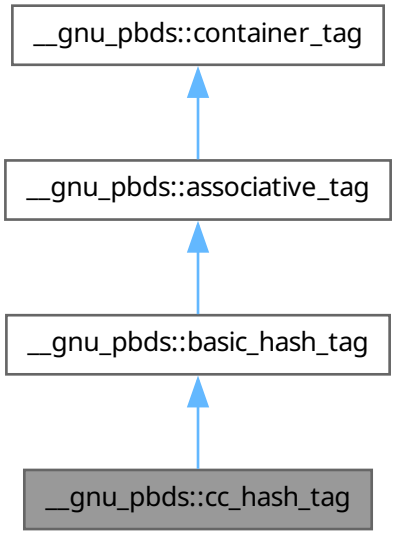
Constructor taking __iterators to a range of value_types and some policy objects The value_types between first_it and last_it will be inserted into the container object. r_hash_fn will be copied by the hash_fn object of the container object, r_eq_fn will be copied by the eq_fn object of the container object, r_comb_hash_fn will be copied by the comb_hash_fn object of the container object, and r_resize_policy will be copied by the resize_policy object of the container object.
The documentation for this class was generated from the following file:

- assoc_container.hpp

5.269 __gnu_pbds::cc_hash_tag Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for __gnu_pbds::cc_hash_tag:



5.269.1 Detailed Description

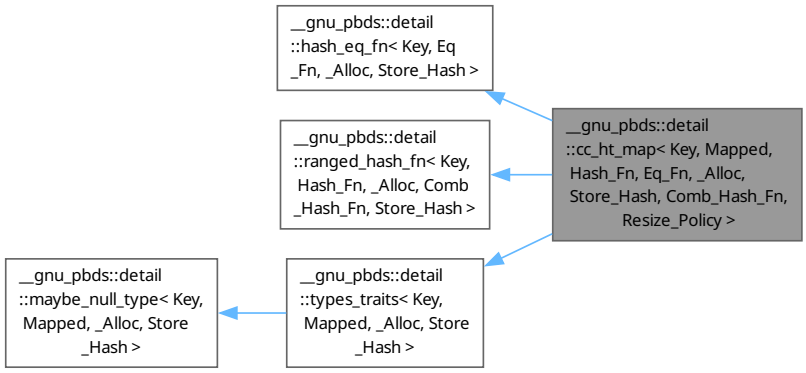
Collision-chaining hash.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

5.270 __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy > Class Template Reference

`#include <cc_ht_map_.hpp>`
Inheritance diagram for __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb↩

_Hash_Fn, Resize_Policy >:



Public Types

- enum { **store_hash** }
- typedef _Alloc **allocator_type**
- typedef Comb_Hash_Fn **comb_hash_fn**
- typedef const_iterator_ **const_iterator**
- typedef traits_base::const_pointer **const_pointer**
- typedef traits_base::const_reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef Eq_Fn **eq_fn**
- typedef Hash_Fn **hash_fn**
- typedef iterator_ **iterator**
- typedef traits_base::key_const_pointer **key_const_pointer**
- typedef traits_base::key_const_reference **key_const_reference**
- typedef traits_base::key_pointer **key_pointer**
- typedef traits_base::key_reference **key_reference**
- typedef traits_base::key_type **key_type**
- typedef traits_base::mapped_const_pointer **mapped_const_pointer**
- typedef traits_base::mapped_const_reference **mapped_const_reference**
- typedef traits_base::mapped_pointer **mapped_pointer**
- typedef traits_base::mapped_reference **mapped_reference**
- typedef traits_base::mapped_type **mapped_type**
- typedef __nothrowcopy::indicator **no_throw_indicator**
- typedef point_const_iterator_ **point_const_iterator**
- typedef point_iterator_ **point_iterator**
- typedef traits_base::pointer **pointer**
- typedef traits_base::reference **reference**
- typedef Resize_Policy **resize_policy**
- typedef _Alloc::size_type **size_type**
- typedef integral_constant< int, Store_Hash > **store_extra**
- typedef stored_data< value_type, size_type, Store_Hash > **stored_data_type**
- typedef traits_base::value_type **value_type**

Public Member Functions

- **cc_ht_map** (const cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy > &)
- **cc_ht_map** (const Hash_Fn &)
- **cc_ht_map** (const Hash_Fn &, const Eq_Fn &)
- **cc_ht_map** (const Hash_Fn &, const Eq_Fn &, const Comb_Hash_Fn &)
- **cc_ht_map** (const Hash_Fn &, const Eq_Fn &, const Comb_Hash_Fn &, const Resize_Policy &)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- template<typename It>
  void **copy_from_range** (It, It)
- bool empty () const
- iterator **end** ()
- const_iterator **end** () const
- bool **erase** (key_const_reference)
- template<typename Pred>
  size_type **erase_if** (Pred)
- point_iterator **find** (key_const_reference)
- point_const_iterator **find** (key_const_reference) const
- point_iterator **find_end** ()
- point_const_iterator **find_end** () const
- Comb_Hash_Fn & get_comb_hash_fn ()
- const Comb_Hash_Fn & get_comb_hash_fn () const
- Eq_Fn & get_eq_fn ()
- const Eq_Fn & get_eq_fn () const
- Hash_Fn & get_hash_fn ()
- const Hash_Fn & get_hash_fn () const
- Resize_Policy & get_resize_policy ()
- const Resize_Policy & get_resize_policy () const
- void **initialize** ()
- std::pair< point_iterator, bool > **insert** (const_reference r_val)
- size_type **max_size** () const
- mapped_reference **operator[ ]** (key_const_reference r_key)
- size_type **size** () const
- void **swap** (cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy > &)

Public Attributes

- no_throw_indicator **m_no_throw_copies_indicator**
- store_extra **m_store_extra_indicator**

Friends

- class **const_iterator_**
- class **iterator_**

5.270.1 Detailed Description

template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
class __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >

A collision-chaining hash-based container.

Template Parameters

Key	Key type.
Mapped	Map type.
Hash_Fn	Hashing functor. Default is __gnu_cxx::hash.
Eq_Fn	Equal functor. Default std::equal_to<Key>
_Alloc	Allocator type.
Store_Hash	If key type stores extra metadata. Defaults to false.
Comb_Hash_Fn	Combining hash functor. If Hash_Fn is not null_type, then this is the ranged-hash functor; otherwise, this is the range-hashing functor. XXX(See Design::Hash-Based Containers::Hash Policies.) Default direct_mask_range_hashing.
Resize_Policy	Resizes hash. Defaults to hash_standard_resize_policy, using hash_exponential_size_policy and hash_load_check_resize_trigger.

Bases are: detail::hash_eq_fn, Resize_Policy, detail::ranged_hash_fn, detail::types_traits. (Optional: detail::debug_↩ map_base.)

5.270.2 Member Enumeration Documentation

anonymous enum

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
anonymous enum
```

Value stores hash, true or false.

5.270.3 Member Function Documentation

empty()

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
bool __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_↩
Fn, Resize_Policy >::empty () const  [inline], [nodiscard]
```

True if size() == 0.

get_comb_hash_fn() [1/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
Comb_Hash_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Hash_Fn, Resize_Policy >::get_comb_hash_fn ()
```

Return current comb_hash_fn.

get_comb_hash_fn() [2/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
const Comb_Hash_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_↩
Hash, Comb_Hash_Fn, Resize_Policy >::get_comb_hash_fn () const
```

Return current const comb_hash_fn.

get_eq_fn() [1/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
Eq_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_↩
Hash_Fn, Resize_Policy >::get_eq_fn ()
```

Return current eq_fn.

get_eq_fn() [2/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
const Eq_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Hash_Fn, Resize_Policy >::get_eq_fn () const
```

Return current const eq_fn.

get_hash_fn() [1/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
Hash_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_↩
Hash_Fn, Resize_Policy >::get_hash_fn ()
```

Return current hash_fn.

get_hash_fn() [2/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
const Hash_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Hash_Fn, Resize_Policy >::get_hash_fn () const
```

Return current const hash_fn.

get_resize_policy() [1/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
Resize_Policy & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Hash_Fn, Resize_Policy >::get_resize_policy ()
```

Return current resize_policy.

get_resize_policy() [2/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
const Resize_Policy & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_↩
Hash, Comb_Hash_Fn, Resize_Policy >::get_resize_policy () const
```

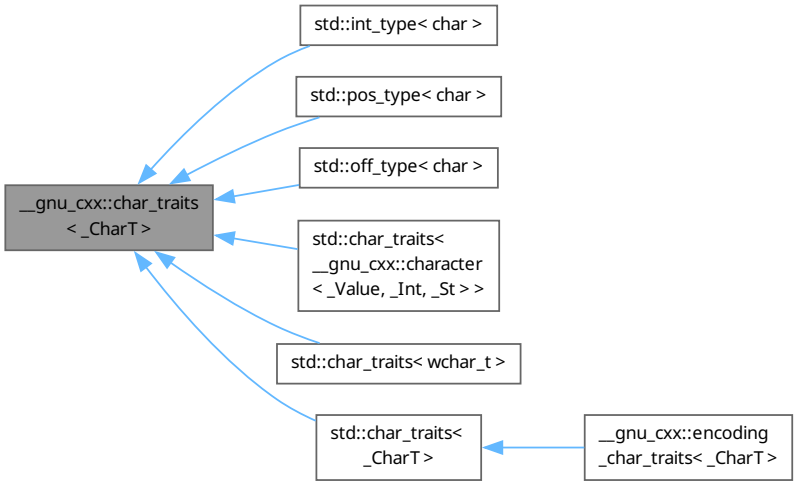
Return current const resize_policy.

The documentation for this class was generated from the following file:

- cc_ht_map_.hpp

5.271 __gnu_cxx::char_traits< _CharT > Struct Template Reference

`#include <char_traits.h>`
Inheritance diagram for __gnu_cxx::char_traits< _CharT >:



Public Types

- typedef _CharT **char_type**
- using **comparison_category**
- typedef _Char_types< _CharT >::int_type **int_type**
- typedef _Char_types< _CharT >::off_type **off_type**
- typedef _Char_types< _CharT >::pos_type **pos_type**
- typedef _Char_types< _CharT >::state_type **state_type**

Static Public Member Functions

- static constexpr void **assign** (char_type &__c1, const char_type &__c2)
- static constexpr char_type * **assign** (char_type *__s, std::size_t __n, char_type __a)
- static constexpr int **compare** (const char_type *__s1, const char_type *__s2, std::size_t __n)
- static constexpr char_type * **copy** (char_type *__s1, const char_type *__s2, std::size_t __n)
- static constexpr int_type **eof** ()
- static constexpr bool **eq** (const char_type &__c1, const char_type &__c2)
- static constexpr bool **eq_int_type** (const int_type &__c1, const int_type &__c2)
- static constexpr const char_type * **find** (const char_type *__s, std::size_t __n, const char_type &__a)
- static constexpr std::size_t **length** (const char_type *__s)
- static constexpr bool **lt** (const char_type &__c1, const char_type &__c2)
- static constexpr char_type * **move** (char_type *__s1, const char_type *__s2, std::size_t __n)
- static constexpr int_type **not_eof** (const int_type &__c)
- static constexpr char_type **to_char_type** (const int_type &__c)
- static constexpr int_type **to_int_type** (const char_type &__c)

5.271.1 Detailed Description

template<typename _CharT>
struct __gnu_cxx::char_traits< _CharT >

Base class used to implement std::char_traits.

Note

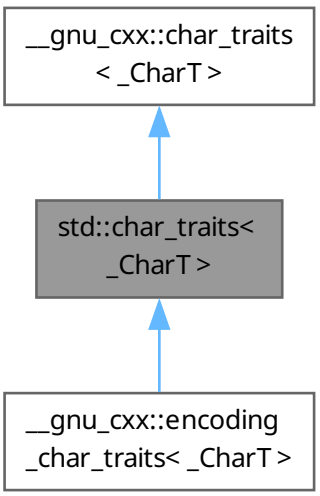
For any given actual character type, this definition is probably wrong. (Most of the member functions are likely to be right, but the int_type and state_type typedefs, and the eof() member function, are likely to be wrong.) The reason this class exists is so users can specialize it. Classes in namespace std may not be specialized for fundamental types, but classes in namespace __gnu_cxx may be.

See https://gcc.gnu.org/onlinedocs/libstdc++/manual/strings.html#strings.string.character_types for advice on how to make use of this class for *unusual* character types. Also, check out include/ext/pod_char_traits.h.
The documentation for this struct was generated from the following file:

- char_traits.h

5.272 std::char_traits< _CharT > Struct Template Reference

`#include <char_traits.h>`
Inheritance diagram for std::char_traits< _CharT >:



Public Types

- typedef _CharT **char_type**
- using **comparison_category**
- typedef _Char_types< _CharT >::int_type **int_type**
- typedef _Char_types< _CharT >::off_type **off_type**
- typedef _Char_types< _CharT >::pos_type **pos_type**
- typedef _Char_types< _CharT >::state_type **state_type**

Static Public Member Functions

- static constexpr void **assign** (char_type &__c1, const char_type &__c2)
- static constexpr char_type * **assign** (char_type *__s, std::size_t __n, char_type __a)
- static constexpr int **compare** (const char_type *__s1, const char_type *__s2, std::size_t __n)
- static constexpr char_type * **copy** (char_type *__s1, const char_type *__s2, std::size_t __n)
- static constexpr int_type **eof** ()
- static constexpr bool **eq** (const char_type &__c1, const char_type &__c2)
- static constexpr bool **eq_int_type** (const int_type &__c1, const int_type &__c2)
- static constexpr const char_type * **find** (const char_type *__s, std::size_t __n, const char_type &__a)
- static constexpr std::size_t **length** (const char_type *__s)
- static constexpr bool **lt** (const char_type &__c1, const char_type &__c2)
- static constexpr char_type * **move** (char_type *__s1, const char_type *__s2, std::size_t __n)
- static constexpr int_type **not_eof** (const int_type &__c)
- static constexpr char_type **to_char_type** (const int_type &__c)
- static constexpr int_type **to_int_type** (const char_type &__c)

5.272.1 Detailed Description

template<**typename _CharT**>
struct std::char_traits< **_CharT** >

Basis for explicit traits specializations.

Note

For any given actual character type, this definition is probably wrong. Since this is just a thin wrapper around __gnu_cxx::char_traits, it is possible to achieve a more appropriate definition by specializing __gnu_cxx::char_traits.

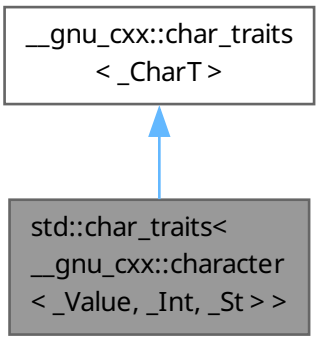
See https://gcc.gnu.org/onlinedocs/libstdc++/manual/strings.html#strings.string.character_types for advice on how to make use of this class for *unusual* character types. Also, check out include/ext/pod_char_traits.h.
The documentation for this struct was generated from the following file:

- char_traits.h

5.273 std::char_traits< __gnu_cxx::character< _Value, _Int, _St > > Struct Template Reference

`#include <pod_char_traits.h>`

Inheritance diagram for std::char_traits< __gnu_cxx::character< _Value, _Int, _St > >:



Public Types

- typedef __gnu_cxx::character< _Value, _Int, _St > **char_type**
- typedef __gnu_cxx::character< _Value, _Int, _St > **char_type**
- using **comparison_category**
- using **comparison_category**
- typedef _Char_types< __gnu_cxx::character< _Value, _Int, _St > >::int_type **int_type**
- typedef char_type::int_type **int_type**
- typedef _Char_types< __gnu_cxx::character< _Value, _Int, _St > >::off_type **off_type**
- typedef streamoff **off_type**
- typedef _Char_types< __gnu_cxx::character< _Value, _Int, _St > >::pos_type **pos_type**
- typedef fpos< state_type > **pos_type**
- typedef _Char_types< __gnu_cxx::character< _Value, _Int, _St > >::state_type **state_type**
- typedef char_type::state_type **state_type**

Static Public Member Functions

- static constexpr void **assign** (char_type &__c1, const char_type &__c2)
- static constexpr char_type * **assign** (char_type *__s, std::size_t __n, char_type __a)
- static constexpr void **assign** (char_type &__c1, const char_type &__c2)
- static void **assign** (char_type &__c1, const char_type &__c2)
- static char_type * **assign** (char_type *__s, size_t __n, char_type __a)
- static constexpr char_type * **assign** (char_type *__s, std::size_t __n, char_type __a)
- static constexpr int **compare** (const char_type *__s1, const char_type *__s2, std::size_t __n)
- static int **compare** (const char_type *__s1, const char_type *__s2, size_t __n)
- static constexpr int **compare** (const char_type *__s1, const char_type *__s2, std::size_t __n)
- static constexpr char_type * **copy** (char_type *__s1, const char_type *__s2, std::size_t __n)
- static char_type * **copy** (char_type *__s1, const char_type *__s2, size_t __n)
- static constexpr char_type * **copy** (char_type *__s1, const char_type *__s2, std::size_t __n)
- static constexpr int_type **eof** ()
- static int_type **eof** ()
- static constexpr bool **eq** (const char_type &__c1, const char_type &__c2)

- static constexpr bool **eq** (const char_type &__c1, const char_type &__c2)
- static bool **eq** (const char_type &__c1, const char_type &__c2)
- static constexpr bool **eq_int_type** (const int_type &__c1, const int_type &__c2)
- static constexpr bool **eq_int_type** (const int_type &__c1, const int_type &__c2)
- static bool **eq_int_type** (const int_type &__c1, const int_type &__c2)
- static constexpr const char_type * **find** (const char_type *__s, std::size_t __n, const char_type &__a)
- static const char_type * **find** (const char_type *__s, size_t __n, const char_type &__a)
- static constexpr const char_type * **find** (const char_type *__s, std::size_t __n, const char_type &__a)
- static constexpr std::size_t **length** (const char_type *__s)
- static constexpr std::size_t **length** (const char_type *__s)
- static size_t **length** (const char_type *__s)
- static constexpr bool **lt** (const char_type &__c1, const char_type &__c2)
- static constexpr bool **lt** (const char_type &__c1, const char_type &__c2)
- static bool **lt** (const char_type &__c1, const char_type &__c2)
- static constexpr char_type * **move** (char_type *__s1, const char_type *__s2, std::size_t __n)
- static char_type * **move** (char_type *__s1, const char_type *__s2, size_t __n)
- static constexpr char_type * **move** (char_type *__s1, const char_type *__s2, std::size_t __n)
- static constexpr int_type **not_eof** (const int_type &__c)
- static constexpr int_type **not_eof** (const int_type &__c)
- static int_type **not_eof** (const int_type &__c)
- static constexpr char_type **to_char_type** (const int_type &__c)
- static constexpr char_type **to_char_type** (const int_type &__c)
- static char_type **to_char_type** (const int_type &__i)
- static constexpr int_type **to_int_type** (const char_type &__c)
- static constexpr int_type **to_int_type** (const char_type &__c)
- static int_type **to_int_type** (const char_type &__c)

5.273.1 Detailed Description

template<**typename _Value, typename _Int, typename _St**>
struct std::char_traits< **__gnu_cxx::character**< **_Value, _Int, _St** > >

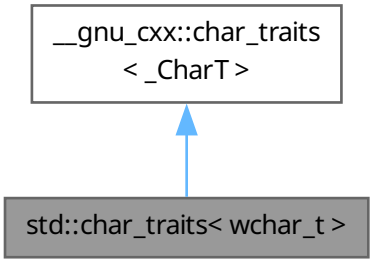
char_traits<__gnu_cxx::character> specialization.
The documentation for this struct was generated from the following file:

- pod_char_traits.h

5.274 std::char_traits< wchar_t > Struct Reference

`#include <char_traits.h>`

Inheritance diagram for std::char_traits< wchar_t >:



Public Types

- typedef wchar_t **char_type**
- typedef wchar_t **char_type**
- using **comparison_category**
- using **comparison_category**
- typedef _Char_types< wchar_t >::int_type **int_type**
- typedef wint_t **int_type**
- typedef _Char_types< wchar_t >::off_type **off_type**
- typedef streamoff **off_type**
- typedef _Char_types< wchar_t >::pos_type **pos_type**
- typedef wstreampos **pos_type**
- typedef _Char_types< wchar_t >::state_type **state_type**
- typedef mbstate_t **state_type**

Static Public Member Functions

- static constexpr void **assign** (char_type &__c1, const char_type &__c2)
- static constexpr char_type * **assign** (char_type *__s, std::size_t __n, char_type __a)
- static constexpr void **assign** (char_type &__c1, const char_type &__c2)
- static constexpr void **assign** (char_type &__c1, const char_type &__c2) noexcept
- static constexpr char_type * **assign** (char_type *__s, size_t __n, char_type __a)
- static constexpr char_type * **assign** (char_type *__s, std::size_t __n, char_type __a)
- static constexpr int **compare** (const char_type *__s1, const char_type *__s2, std::size_t __n)
- static constexpr int **compare** (const char_type *__s1, const char_type *__s2, size_t __n)
- static constexpr int **compare** (const char_type *__s1, const char_type *__s2, std::size_t __n)
- static constexpr char_type * **copy** (char_type *__s1, const char_type *__s2, std::size_t __n)
- static constexpr char_type * **copy** (char_type *__s1, const char_type *__s2, size_t __n)
- static constexpr char_type * **copy** (char_type *__s1, const char_type *__s2, std::size_t __n)
- static constexpr int_type **eof** ()
- static constexpr int_type **eof** () noexcept
- static constexpr bool **eq** (const char_type &__c1, const char_type &__c2)
- static constexpr bool **eq** (const char_type &__c1, const char_type &__c2)
- static constexpr bool **eq** (const char_type &__c1, const char_type &__c2) noexcept

- static constexpr bool **eq_int_type** (const int_type &__c1, const int_type &__c2)
- static constexpr bool **eq_int_type** (const int_type &__c1, const int_type &__c2)
- static constexpr bool **eq_int_type** (const int_type &__c1, const int_type &__c2) noexcept
- static constexpr const char_type * **find** (const char_type *__s, std::size_t __n, const char_type &__a)
- static constexpr const char_type * **find** (const char_type *__s, size_t __n, const char_type &__a)
- static constexpr const char_type * **find** (const char_type *__s, std::size_t __n, const char_type &__a)
- static constexpr std::size_t **length** (const char_type *__s)
- static constexpr std::size_t **length** (const char_type *__s)
- static constexpr size_t **length** (const char_type *__s)
- static constexpr bool **lt** (const char_type &__c1, const char_type &__c2)
- static constexpr bool **lt** (const char_type &__c1, const char_type &__c2)
- static constexpr bool **lt** (const char_type &__c1, const char_type &__c2) noexcept
- static constexpr char_type * **move** (char_type *__s1, const char_type *__s2, std::size_t __n)
- static constexpr char_type * **move** (char_type *__s1, const char_type *__s2, size_t __n)
- static constexpr char_type * **move** (char_type *__s1, const char_type *__s2, std::size_t __n)
- static constexpr int_type **not_eof** (const int_type &__c)
- static constexpr int_type **not_eof** (const int_type &__c)
- static constexpr int_type **not_eof** (const int_type &__c) noexcept
- static constexpr char_type **to_char_type** (const int_type &__c)
- static constexpr char_type **to_char_type** (const int_type &__c)
- static constexpr char_type **to_char_type** (const int_type &__c) noexcept
- static constexpr int_type **to_int_type** (const char_type &__c)
- static constexpr int_type **to_int_type** (const char_type &__c)
- static constexpr int_type **to_int_type** (const char_type &__c) noexcept

5.274.1 Detailed Description

21.1.3.2 char_traits specializations
The documentation for this struct was generated from the following file:

- char_traits.h

5.275 __gnu_cxx::character< _Value, _Int, _St > Struct Template Reference

`#include <pod_char_traits.h>`

Public Types

- typedef character< _Value, _Int, _St > **char_type**
- typedef _Int **int_type**
- typedef _St **state_type**
- typedef _Value **value_type**

Static Public Member Functions

- template<typename V2>
  static char_type **from** (const V2 &v)
- template<typename V2>
  static V2 **to** (const char_type &c)

Public Attributes

- value_type **value**

5.275.1 Detailed Description

template<typename _Value, typename _Int, typename _St = std::mbstate_t>
struct __gnu_cxx::character< _Value, _Int, _St >

A POD class that serves as a character abstraction class.
The documentation for this struct was generated from the following file:

- pod_char_traits.h

5.276 std::chi_squared_distribution< _RealType > Class Template Reference

`#include <random>`

Classes

- struct param_type

Public Types

- typedef _RealType result_type

Public Member Functions

- **chi_squared_distribution** (_RealType __n)
- **chi_squared_distribution** (const param_type &__p)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- result_type max () const
- result_type min () const
- _RealType **n** () const
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- template<typename _RealType1, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::chi_squared_distribution< _RealType1 > &__x)
- bool operator== (const chi_squared_distribution &__d1, const chi_squared_distribution &__d2)
- template<typename _RealType1, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::chi_squared_distribution< _RealType1 > &__x)

5.276.1 Detailed Description

template<typename _RealType = double>
class std::chi_squared_distribution< _RealType >

A chi_squared_distribution random number distribution.
The formula for the normal probability mass function is $p(x|n) = \frac{x^{(n/2)-1}e^{-x/2}}{\Gamma(n/2)2^{n/2}}$

Since

C++11

5.276.2 Member Typedef Documentation

result_type

```
template<typename _RealType = double>
typedef _RealType std::chi_squared_distribution< _RealType >::result_type
```

The type of the range of the distribution.

5.276.3 Member Function Documentation

max()

```
template<typename _RealType = double>
result_type std::chi_squared_distribution< _RealType >::max () const  [inline]
```

Returns the least upper bound value of the distribution.

min()

```
template<typename _RealType = double>
result_type std::chi_squared_distribution< _RealType >::min () const  [inline]
```

Returns the greatest lower bound value of the distribution.

operator()()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::chi_squared_distribution< _RealType >::operator() (
            _UniformRandomNumberGenerator & __urng)  [inline]
```

Generating functions.

param() [1/2]

```
template<typename _RealType = double>
param_type std::chi_squared_distribution< _RealType >::param () const  [inline]
```

Returns the parameter set of the distribution.

param() [2/2]

```
template<typename _RealType = double>
void std::chi_squared_distribution< _RealType >::param (
            const param_type & __param)  [inline]
```

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

reset()

```
template<typename _RealType = double>
void std::chi_squared_distribution< _RealType >::reset ()  [inline]
```

Resets the distribution state.

5.276.4 Friends And Related Symbol Documentation

operator<<

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::chi_squared_distribution< _RealType1 > & __x)  [friend]
```

Inserts a chi_squared_distribution random number distribution __x into the output stream __os.

Parameters

__os	An output stream.
__x	A chi_squared_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

operator==

```
template<typename _RealType = double>
bool operator== (
            const chi_squared_distribution< _RealType > & __d1,
            const chi_squared_distribution< _RealType > & __d2)  [friend]
```

Return true if two Chi-squared distributions have the same parameters and the sequences that would be generated are equal.

operator>>

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            std::chi_squared_distribution< _RealType1 > & __x)  [friend]
```

Extracts a chi_squared_distribution random number distribution __x from the input stream __is.

Parameters

__is	An input stream.
__x	A chi_squared_distribution random number generator engine.

Returns

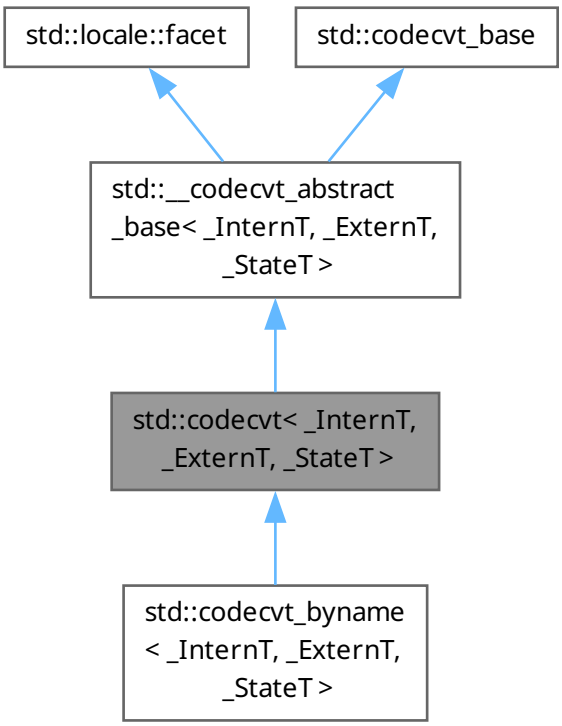
The input stream with __x extracted or in an error state.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

5.277 std::codecvt< _InternT, _ExternT, _StateT > Class Template Reference

`#include <codecvt.h>`
Inheritance diagram for std::codecvt< _InternT, _ExternT, _StateT >:



Public Types

- typedef _ExternT **extern_type**
- typedef _InternT **intern_type**
- typedef codecvt_base::result **result**
- typedef _StateT **state_type**

Public Member Functions

- **codecvt** (__c_locale __cloc, size_t __refs=0)
- **codecvt** (size_t __refs=0)

- bool **always_noconv** () const throw ()
- int **encoding** () const throw ()
- result in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- int **length** (state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max) const
- int **max_length** () const throw ()
- result out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const

Static Public Attributes

- static locale::id **id**

Protected Member Functions

- virtual bool do_always_noconv () const throw ()
- virtual int do_encoding () const throw ()
- virtual result do_in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- virtual int do_length (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const
- virtual int do_max_length () const throw ()
- virtual result do_out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result do_unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale **_M_c_locale_codecvt**

5.277.1 Detailed Description

template<typename _InternT, typename _ExternT, typename _StateT>
class std::codecvt< _InternT, _ExternT, _StateT >

Primary class template codecvt.
NB: Generic, mostly useless implementation.

5.277.2 Member Function Documentation

do_always_noconv()

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual bool std::codecvt< _InternT, _ExternT, _StateT >::do_always_noconv () const throw ( )
[protected], [virtual]
```

Implements std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >.

do_encoding()

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual int std::codecvt< _InternT, _ExternT, _StateT >::do_encoding () const throw ( )   [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >.

do_in()

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual result std::codecvt< _InternT, _ExternT, _StateT >::do_in (
            state_type & __state,
            const extern_type * __from,
            const extern_type * __from_end,
            const extern_type *& __from_next,
            intern_type * __to,
            intern_type * __to_end,
            intern_type *& __to_next) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >.

do_length()

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual int std::codecvt< _InternT, _ExternT, _StateT >::do_length (
            state_type & ,
            const extern_type * __from,
            const extern_type * __end,
            size_t __max) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >.

do_max_length()

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual int std::codecvt< _InternT, _ExternT, _StateT >::do_max_length () const throw ( )   [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >.

do_out()

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual result std::codecvt< _InternT, _ExternT, _StateT >::do_out (
            state_type & __state,
            const intern_type * __from,
            const intern_type * __from_end,
            const intern_type *& __from_next,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [protected], [virtual]
```

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This function is a hook for derived classes to change the value returned.

See also

out for more information.

Implements std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >.

do_unshift()

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual result std::codecvt< _InternT, _ExternT, _StateT >::do_unshift (
            state_type & __state,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >.

in()

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::in (
            state_type & __state,
            const extern_type * __from,
            const extern_type * __from_end,
            const extern_type *& __from_next,
            intern_type * __to,
            intern_type * __to_end,
            intern_type *& __to_next) const  [inline], [inherited]
```

Convert from external to internal character set.

Converts input string of extern_type to output string of intern_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do_in.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

out()

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::out (
            state_type & __state,
```

```
            const intern_type * __from,
            const intern_type * __from_end,
            const intern_type *& __from_next,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [inline], [inherited]
```

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This is analogous to wcsrtombs. It does this by calling codecvt::do_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

References do_out().

unshift()

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::unshift (
            state_type & __state,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [inline], [inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of codecvt_base::result. If the state could be reset and data written, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the output has insufficient space, returns codecvt_base::partial. Otherwise the reset failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

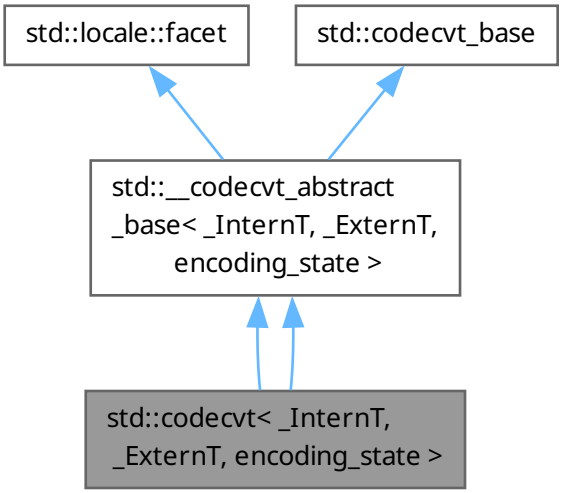
codecvt_base::result.

The documentation for this class was generated from the following file:

- codecvt.h

5.278 std::codecvt< _InternT, _ExternT, encoding_state > Class Template Reference

`#include <codecvt_specializations.h>`
Inheritance diagram for std::codecvt< _InternT, _ExternT, encoding_state >:



Public Types

- typedef state_type::descriptor_type **descriptor_type**
- typedef _ExternT **extern_type**
- typedef _ExternT **extern_type**
- typedef _InternT **intern_type**
- typedef _InternT **intern_type**
- typedef codecvt_base::result **result**
- typedef codecvt_base::result **result**
- typedef encoding_state **state_type**
- typedef __gnu_cxx::encoding_state **state_type**

Public Member Functions

- **codecvt** (__c_locale __cloc, size_t __refs=0)
- **codecvt** (size_t __refs=0)
- **codecvt** (size_t __refs=0)
- **codecvt** (state_type &__enc, size_t __refs=0)
- bool **always_noconv** () const throw ()
- bool **always_noconv** () const throw ()
- int **encoding** () const throw ()
- int **encoding** () const throw ()
- result in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- result in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- int **length** (state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max) const
- int **length** (state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max) const
- int **max_length** () const throw ()
- int **max_length** () const throw ()
- result out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const

Static Public Attributes

- static locale::id **id**
- static locale::id **id**

Protected Member Functions

- virtual bool do_always_noconv () const throw ()
- virtual bool do_always_noconv () const throw ()
- virtual int do_encoding () const throw ()
- virtual int do_encoding () const throw ()
- virtual result do_in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- virtual result do_in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- virtual int do_length (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const
- virtual int do_length (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const
- virtual int do_max_length () const throw ()
- virtual int do_max_length () const throw ()
- virtual result do_out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result do_out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result do_unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to↩_next) const
- virtual result do_unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to↩_next) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale **_M_c_locale_codecvt**

5.278.1 Detailed Description

template<typename _InternT, typename _ExternT>
class std::codecvt< _InternT, _ExternT, encoding_state >

codecvt<InternT, _ExternT, encoding_state> specialization.

5.278.2 Member Function Documentation

do_always_noconv() [1/2]

```
virtual bool std::codecvt< _InternT, _ExternT, encoding_state >::do_always_noconv () const throw
( )   [protected], [virtual]
```

Implements std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >.

do_always_noconv() [2/2]

```
template<typename _InternT, typename _ExternT>
bool std::codecvt< _InternT, _ExternT, encoding_state >::do_always_noconv () const throw ( )
[protected], [virtual]
```

Implements std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >.

do_encoding() [1/2]

```
virtual int std::codecvt< _InternT, _ExternT, encoding_state >::do_encoding () const throw ( )
[protected], [virtual]
```

Implements std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >.

do_encoding() [2/2]

```
template<typename _InternT, typename _ExternT>
int std::codecvt< _InternT, _ExternT, encoding_state >::do_encoding () const throw ( )   [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >.

do_in() [1/2]

```
virtual result std::codecvt< _InternT, _ExternT, encoding_state >::do_in (
            state_type & __state,
            const extern_type * __from,
            const extern_type * __from_end,
            const extern_type *& __from_next,
            intern_type * __to,
            intern_type * __to_end,
            intern_type *& __to_next) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >.

do_in() [2/2]

```
template<typename _InternT, typename _ExternT>
codecvt_base::result std::codecvt< _InternT, _ExternT, encoding_state >::do_in (
            state_type & __state,
            const extern_type * __from,
            const extern_type * __from_end,
            const extern_type *& __from_next,
            intern_type * __to,
            intern_type * __to_end,
            intern_type *& __to_next) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >.

do_length() [1/2]

```
virtual int std::codecvt< _InternT, _ExternT, encoding_state >::do_length (
            state_type & ,
            const extern_type * __from,
            const extern_type * __end,
            size_t __max) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >.

do_length() [2/2]

```
template<typename _InternT, typename _ExternT>
int std::codecvt< _InternT, _ExternT, encoding_state >::do_length (
            state_type & ,
            const extern_type * __from,
            const extern_type * __end,
            size_t __max) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >.

do_max_length() [1/2]

```
virtual int std::codecvt< _InternT, _ExternT, encoding_state >::do_max_length () const throw ( )
[protected], [virtual]
```

Implements std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >.

do_max_length() [2/2]

```
template<typename _InternT, typename _ExternT>
int std::codecvt< _InternT, _ExternT, encoding_state >::do_max_length () const throw ( )   [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >.

do_out() [1/2]

```
virtual result std::codecvt< _InternT, _ExternT, encoding_state >::do_out (
            state_type & __state,
            const intern_type * __from,
            const intern_type * __from_end,
            const intern_type *& __from_next,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [protected], [virtual]
```

Convert from internal to external character set.
Converts input string of intern_type to output string of extern_type. This function is a hook for derived classes to change the value returned.

See also

out for more information.

Implements std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >.

do_out() [2/2]

```
template<typename _InternT, typename _ExternT>
codecvt_base::result std::codecvt< _InternT, _ExternT, encoding_state >::do_out (
            state_type & __state,
            const intern_type * __from,
            const intern_type * __from_end,
            const intern_type *& __from_next,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [protected], [virtual]
```

Convert from internal to external character set.
Converts input string of intern_type to output string of extern_type. This function is a hook for derived classes to change the value returned.

See also

out for more information.

Implements std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >.

do_unshift() [1/2]

```
virtual result std::codecvt< _InternT, _ExternT, encoding_state >::do_unshift (
            state_type & __state,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >.

do_unshift() [2/2]

```
template<typename _InternT, typename _ExternT>
codecvt_base::result std::codecvt< _InternT, _ExternT, encoding_state >::do_unshift (
            state_type & __state,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >.

in() [1/2]

```
result std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >::in (
             state_type & __state,
             const extern_type * __from,
             const extern_type * __from_end,
             const extern_type *& __from_next,
             intern_type * __to,
             intern_type * __to_end,
             intern_type *& __to_next) const  [inline], [inherited]
```

Convert from external to internal character set.

Converts input string of extern_type to output string of intern_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do_in.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

in() [2/2]

```
result std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >::in (
             state_type & __state,
             const extern_type * __from,
             const extern_type * __from_end,
             const extern_type *& __from_next,
             intern_type * __to,
             intern_type * __to_end,
             intern_type *& __to_next) const  [inline]
```

Convert from external to internal character set.

Converts input string of extern_type to output string of intern_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do_in.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.
The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

out() [1/2]

```
result std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >::out (
             state_type & __state,
             const intern_type * __from,
             const intern_type * __from_end,
             const intern_type *& __from_next,
             extern_type * __to,
             extern_type * __to_end,
             extern_type *& __to_next) const  [inline], [inherited]
```

Convert from internal to external character set.
Converts input string of intern_type to output string of extern_type. This is analogous to wcsrtombs. It does this by calling codecvt::do_out.
The source and destination character sets are determined by the facet's locale, internal and external types.
The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.
The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.
The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

out() [2/2]

```
result std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >::out (
            state_type & __state,
            const intern_type * __from,
            const intern_type * __from_end,
            const intern_type *& __from_next,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [inline]
```

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This is analogous to wcsrtombs. It does this by calling codecvt::do_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

unshift() [1/2]

```
result std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >::unshift (
            state_type & __state,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [inline], [inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do_unshift().
For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.
The source and destination character sets are determined by the facet's locale, internal and external types.
The result returned is a member of codecvt_base::result. If the state could be reset and data written, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the output has insufficient space, returns codecvt_base::partial. Otherwise the reset failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

unshift() [2/2]

```
result std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >::unshift (
            state_type & __state,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [inline]
```

Reset conversion state.
Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do_unshift().
For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.
The source and destination character sets are determined by the facet's locale, internal and external types.
The result returned is a member of codecvt_base::result. If the state could be reset and data written, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the output has insufficient space, returns codecvt_base::partial. Otherwise the reset failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

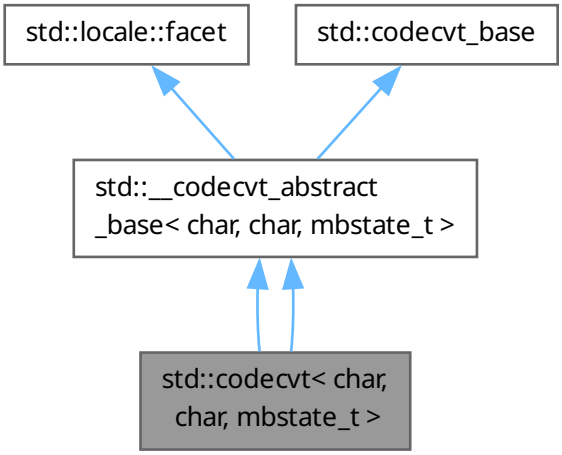
codecvt_base::result.

The documentation for this class was generated from the following file:

- codecvt_specializations.h

5.279 std::codecvt< char, char, mbstate_t > Class Reference

`#include <codecvt.h>`
Inheritance diagram for std::codecvt< char, char, mbstate_t >:



Public Types

- typedef char **extern_type**
- typedef char **extern_type**
- typedef char **intern_type**
- typedef char **intern_type**
- typedef codecvt_base::result **result**
- typedef mbstate_t **state_type**
- typedef mbstate_t **state_type**

Public Member Functions

- **codecvt** (__c_locale __cloc, size_t __refs=0)
- **codecvt** (__c_locale __cloc, size_t __refs=0)
- **codecvt** (size_t __refs=0)
- **codecvt** (size_t __refs=0)
- bool **always_noconv** () const throw ()
- bool **always_noconv** () const throw ()
- int **encoding** () const throw ()
- int **encoding** () const throw ()
- result in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- result in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- int **length** (state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max) const
- int **length** (state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max) const

- int **max_length** () const throw ()
- int **max_length** () const throw ()
- result out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const

Static Public Attributes

- static locale::id **id**
- static locale::id **id**

Protected Member Functions

- virtual bool do_always_noconv () const throw ()
- virtual bool do_always_noconv () const throw ()
- virtual int do_encoding () const throw ()
- virtual int do_encoding () const throw ()
- virtual result do_in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- virtual result do_in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- virtual int do_length (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const
- virtual int do_length (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const
- virtual int do_max_length () const throw ()
- virtual int do_max_length () const throw ()
- virtual result do_out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result do_out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result do_unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to↩ _next) const
- virtual result do_unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to↩ _next) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale **_M_c_locale_codecvt**
- __c_locale **_M_c_locale_codecvt**

Friends

- class **messages**< **char** >

5.279.1 Detailed Description

class codecvt<char, char, mbstate_t> specialization.

5.279.2 Member Function Documentation

do_always_noconv() [1/2]

```
virtual bool std::codecvt< char, char, mbstate_t >::do_always_noconv () const throw ( )   [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< char, char, mbstate_t >.

do_always_noconv() [2/2]

```
virtual bool std::codecvt< char, char, mbstate_t >::do_always_noconv () const throw ( )   [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< char, char, mbstate_t >.

do_encoding() [1/2]

```
virtual int std::codecvt< char, char, mbstate_t >::do_encoding () const throw ( )   [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< char, char, mbstate_t >.

do_encoding() [2/2]

```
virtual int std::codecvt< char, char, mbstate_t >::do_encoding () const throw ( )   [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< char, char, mbstate_t >.

do_in() [1/2]

```
virtual result std::codecvt< char, char, mbstate_t >::do_in (
            state_type & __state,
            const extern_type * __from,
            const extern_type * __from_end,
            const extern_type *& __from_next,
            intern_type * __to,
            intern_type * __to_end,
            intern_type *& __to_next) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< char, char, mbstate_t >.

do_in() [2/2]

```
virtual result std::codecvt< char, char, mbstate_t >::do_in (
            state_type & __state,
            const extern_type * __from,
```

```
            const extern_type * __from_end,
            const extern_type *& __from_next,
            intern_type * __to,
            intern_type * __to_end,
            intern_type *& __to_next) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< char, char, mbstate_t >.

do_length() [1/2]

```
virtual int std::codecvt< char, char, mbstate_t >::do_length (
            state_type & ,
            const extern_type * __from,
            const extern_type * __end,
            size_t __max) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< char, char, mbstate_t >.

do_length() [2/2]

```
virtual int std::codecvt< char, char, mbstate_t >::do_length (
            state_type & ,
            const extern_type * __from,
            const extern_type * __end,
            size_t __max) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< char, char, mbstate_t >.

do_max_length() [1/2]

```
virtual int std::codecvt< char, char, mbstate_t >::do_max_length () const throw ( )  [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< char, char, mbstate_t >.

do_max_length() [2/2]

```
virtual int std::codecvt< char, char, mbstate_t >::do_max_length () const throw ( )  [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< char, char, mbstate_t >.

do_out() [1/2]

```
virtual result std::codecvt< char, char, mbstate_t >::do_out (
            state_type & __state,
            const intern_type * __from,
            const intern_type * __from_end,
            const intern_type *& __from_next,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [protected], [virtual]
```

Convert from internal to external character set.
Converts input string of intern_type to output string of extern_type. This function is a hook for derived classes to change the value returned.

See also

out for more information.

Implements std::__codecvt_abstract_base< char, char, mbstate_t >.

do_out() [2/2]

```
virtual result std::codecvt< char, char, mbstate_t >::do_out (
            state_type & __state,
            const intern_type * __from,
            const intern_type * __from_end,
            const intern_type *& __from_next,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [protected], [virtual]
```

Convert from internal to external character set.
Converts input string of intern_type to output string of extern_type. This function is a hook for derived classes to change the value returned.

See also

out for more information.

Implements std::__codecvt_abstract_base< char, char, mbstate_t >.

do_unshift() [1/2]

```
virtual result std::codecvt< char, char, mbstate_t >::do_unshift (
            state_type & __state,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< char, char, mbstate_t >.

do_unshift() [2/2]

```
virtual result std::codecvt< char, char, mbstate_t >::do_unshift (
            state_type & __state,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< char, char, mbstate_t >.

in() [1/2]

```
result std::__codecvt_abstract_base< char, char, mbstate_t >::in (
            state_type & __state,
            const extern_type * __from,
            const extern_type * __from_end,
            const extern_type *& __from_next,
            intern_type * __to,
            intern_type * __to_end,
            intern_type *& __to_next) const  [inline], [inherited]
```

Convert from external to internal character set.
Converts input string of extern_type to output string of intern_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do_in.
The source and destination character sets are determined by the facet's locale, internal and external types.
The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.
The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

in() [2/2]

```
result std::__codecvt_abstract_base< char, char, mbstate_t >::in (
             state_type & __state,
             const extern_type * __from,
             const extern_type * __from_end,
             const extern_type *& __from_next,
             intern_type * __to,
             intern_type * __to_end,
             intern_type *& __to_next) const  [inline]
```

Convert from external to internal character set.

Converts input string of extern_type to output string of intern_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do_in.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

out() [1/2]

```
result std::__codecvt_abstract_base< char, char, mbstate_t >::out (
             state_type & __state,
             const intern_type * __from,
             const intern_type * __from_end,
             const intern_type *& __from_next,
             extern_type * __to,
             extern_type * __to_end,
             extern_type *& __to_next) const  [inline], [inherited]
```

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This is analogous to wcsrtombs. It does this by calling codecvt::do_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

out() [2/2]

```
result std::__codecvt_abstract_base< char, char, mbstate_t >::out (
             state_type & __state,
             const intern_type * __from,
             const intern_type * __from_end,
             const intern_type *& __from_next,
             extern_type * __to,
             extern_type * __to_end,
             extern_type *& __to_next) const  [inline]
```

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This is analogous to wcsrtombs. It does this by calling codecvt::do_out.
The source and destination character sets are determined by the facet's locale, internal and external types.
The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.
The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.
The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

unshift() [1/2]

```
result std::__codecvt_abstract_base< char, char, mbstate_t >::unshift (
            state_type & __state,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [inline], [inherited]
```

Reset conversion state.
Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do_unshift().
For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.
The source and destination character sets are determined by the facet's locale, internal and external types.
The result returned is a member of codecvt_base::result. If the state could be reset and data written, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the output has insufficient space, returns codecvt_base::partial. Otherwise the reset failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

unshift() [2/2]

```
result std::__codecvt_abstract_base< char, char, mbstate_t >::unshift (
            state_type & __state,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [inline]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of codecvt_base::result. If the state could be reset and data written, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the output has insufficient space, returns codecvt_base::partial. Otherwise the reset failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

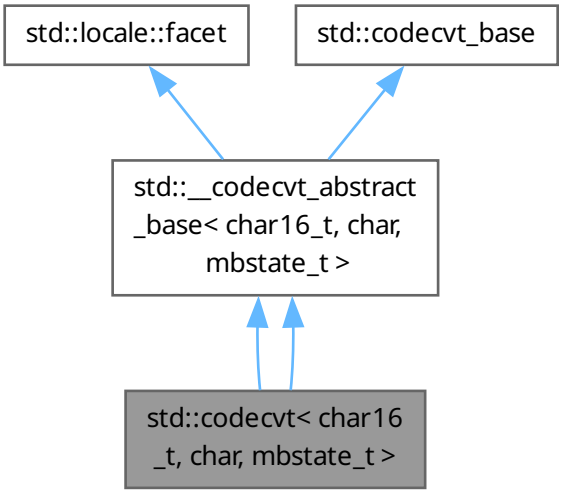
The documentation for this class was generated from the following file:

- codecvt.h

5.280 std::codecvt< char16_t, char, mbstate_t > Class Reference

```
#include <codecvt.h>
```

Inheritance diagram for std::codecvt< char16_t, char, mbstate_t >:



Public Types

- typedef char **extern_type**
- typedef char **extern_type**
- typedef char16_t **intern_type**
- typedef char16_t **intern_type**
- typedef codecvt_base::result **result**
- typedef mbstate_t **state_type**
- typedef mbstate_t **state_type**

Public Member Functions

- **codecvt** (__c_locale __cloc, size_t __refs=0)
- **codecvt** (size_t __refs=0)
- **codecvt** (size_t __refs=0)
- bool **always_noconv** () const throw ()
- bool **always_noconv** () const throw ()
- int **encoding** () const throw ()
- int **encoding** () const throw ()
- result in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- result in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- int **length** (state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max) const
- int **length** (state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max) const
- int **max_length** () const throw ()
- int **max_length** () const throw ()

- result out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const

Static Public Attributes

- static locale::id **id**
- static locale::id **id**

Protected Member Functions

- virtual bool do_always_noconv () const throw ()
- virtual bool do_always_noconv () const throw ()
- virtual int do_encoding () const throw ()
- virtual int do_encoding () const throw ()
- virtual result do_in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- virtual result do_in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- virtual int do_length (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const
- virtual int do_length (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const
- virtual int do_max_length () const throw ()
- virtual int do_max_length () const throw ()
- virtual result do_out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result do_out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result do_unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result do_unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale **_M_c_locale_codecvt**

5.280.1 Detailed Description

Class codecvt<char16_t, char, mbstate_t> specialization.
Converts between UTF-16 and UTF-8.

5.280.2 Member Function Documentation

do_always_noconv() [1/2]

```
virtual bool std::codecvt< char16_t, char, mbstate_t >::do_always_noconv () const throw ( )   [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< char16_t, char, mbstate_t >.

do_always_noconv() [2/2]

```
virtual bool std::codecvt< char16_t, char, mbstate_t >::do_always_noconv () const throw ( )   [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< char16_t, char, mbstate_t >.

do_encoding() [1/2]

```
virtual int std::codecvt< char16_t, char, mbstate_t >::do_encoding () const throw ( )   [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< char16_t, char, mbstate_t >.

do_encoding() [2/2]

```
virtual int std::codecvt< char16_t, char, mbstate_t >::do_encoding () const throw ( )   [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< char16_t, char, mbstate_t >.

do_in() [1/2]

```
virtual result std::codecvt< char16_t, char, mbstate_t >::do_in (
            state_type & __state,
            const extern_type * __from,
            const extern_type * __from_end,
            const extern_type *& __from_next,
            intern_type * __to,
            intern_type * __to_end,
            intern_type *& __to_next) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< char16_t, char, mbstate_t >.

do_in() [2/2]

```
virtual result std::codecvt< char16_t, char, mbstate_t >::do_in (
            state_type & __state,
            const extern_type * __from,
            const extern_type * __from_end,
            const extern_type *& __from_next,
            intern_type * __to,
            intern_type * __to_end,
            intern_type *& __to_next) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< char16_t, char, mbstate_t >.

do_length() [1/2]

```
virtual int std::codecvt< char16_t, char, mbstate_t >::do_length (
            state_type & ,
            const extern_type * __from,
            const extern_type * __end,
            size_t __max) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< char16_t, char, mbstate_t >.

do_length() [2/2]

```
virtual int std::codecvt< char16_t, char, mbstate_t >::do_length (
            state_type & ,
            const extern_type * __from,
            const extern_type * __end,
            size_t __max) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< char16_t, char, mbstate_t >.

do_max_length() [1/2]

```
virtual int std::codecvt< char16_t, char, mbstate_t >::do_max_length () const throw ( )   [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< char16_t, char, mbstate_t >.

do_max_length() [2/2]

```
virtual int std::codecvt< char16_t, char, mbstate_t >::do_max_length () const throw ( )   [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< char16_t, char, mbstate_t >.

do_out() [1/2]

```
virtual result std::codecvt< char16_t, char, mbstate_t >::do_out (
            state_type & __state,
            const intern_type * __from,
            const intern_type * __from_end,
            const intern_type *& __from_next,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [protected], [virtual]
```

Convert from internal to external character set.
Converts input string of intern_type to output string of extern_type. This function is a hook for derived classes to change the value returned.

See also

out for more information.

Implements std::__codecvt_abstract_base< char16_t, char, mbstate_t >.

do_out() [2/2]

```
virtual result std::codecvt< char16_t, char, mbstate_t >::do_out (
            state_type & __state,
            const intern_type * __from,
            const intern_type * __from_end,
            const intern_type *& __from_next,
```

```
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [protected], [virtual]
```

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This function is a hook for derived classes to change the value returned.

See also

out for more information.

Implements std::__codecvt_abstract_base< char16_t, char, mbstate_t >.

do_unshift() [1/2]

```
virtual result std::codecvt< char16_t, char, mbstate_t >::do_unshift (
            state_type & __state,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< char16_t, char, mbstate_t >.

do_unshift() [2/2]

```
virtual result std::codecvt< char16_t, char, mbstate_t >::do_unshift (
            state_type & __state,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< char16_t, char, mbstate_t >.

in() [1/2]

```
result std::__codecvt_abstract_base< char16_t, char, mbstate_t >::in (
            state_type & __state,
            const extern_type * __from,
            const extern_type * __from_end,
            const extern_type *& __from_next,
            intern_type * __to,
            intern_type * __to_end,
            intern_type *& __to_next) const  [inline], [inherited]
```

Convert from external to internal character set.

Converts input string of extern_type to output string of intern_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do_in.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.

Parameters

__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

in() [2/2]

```
result std::__codecvt_abstract_base< char16_t, char, mbstate_t >::in (
             state_type & __state,
             const extern_type * __from,
             const extern_type * __from_end,
             const extern_type *& __from_next,
             intern_type * __to,
             intern_type * __to_end,
             intern_type *& __to_next) const  [inline]
```

Convert from external to internal character set.

Converts input string of extern_type to output string of intern_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do_in.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

out() [1/2]

```
result std::__codecvt_abstract_base< char16_t, char, mbstate_t >::out (
            state_type & __state,
            const intern_type * __from,
            const intern_type * __from_end,
            const intern_type *& __from_next,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [inline], [inherited]
```

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This is analogous to wcsrtombs. It does this by calling codecvt::do_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

out() [2/2]

```
result std::__codecvt_abstract_base< char16_t, char, mbstate_t >::out (
            state_type & __state,
            const intern_type * __from,
            const intern_type * __from_end,
            const intern_type *& __from_next,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [inline]
```

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This is analogous to wcsrtombs. It does this by calling codecvt::do_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.
The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

unshift() [1/2]

```
result std::__codecvt_abstract_base< char16_t, char, mbstate_t >::unshift (
            state_type & __state,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [inline], [inherited]
```

Reset conversion state.
Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do_unshift().
For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.
The source and destination character sets are determined by the facet's locale, internal and external types.
The result returned is a member of codecvt_base::result. If the state could be reset and data written, returns codecvt↩ _base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the output has insufficient space, returns codecvt_base::partial. Otherwise the reset failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

unshift() [2/2]

```
result std::__codecvt_abstract_base< char16_t, char, mbstate_t >::unshift (
            state_type & __state,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [inline]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of codecvt_base::result. If the state could be reset and data written, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the output has insufficient space, returns codecvt_base::partial. Otherwise the reset failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

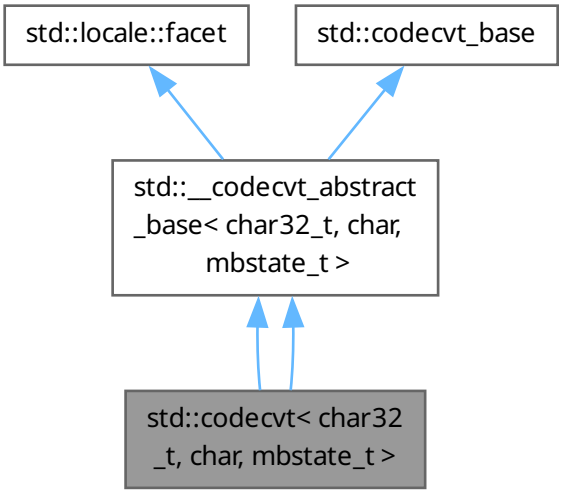
The documentation for this class was generated from the following file:

- codecvt.h

5.281 std::codecvt< char32_t, char, mbstate_t > Class Reference

```
#include <codecvt.h>
```

Inheritance diagram for std::codecvt< char32_t, char, mbstate_t >:



Public Types

- typedef char **extern_type**
- typedef char **extern_type**
- typedef char32_t **intern_type**
- typedef char32_t **intern_type**
- typedef codecvt_base::result **result**
- typedef mbstate_t **state_type**
- typedef mbstate_t **state_type**

Public Member Functions

- **codecvt** (__c_locale __cloc, size_t __refs=0)
- **codecvt** (size_t __refs=0)
- **codecvt** (size_t __refs=0)
- bool **always_noconv** () const throw ()
- bool **always_noconv** () const throw ()
- int **encoding** () const throw ()
- int **encoding** () const throw ()
- result in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- result in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- int **length** (state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max) const
- int **length** (state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max) const
- int **max_length** () const throw ()
- int **max_length** () const throw ()

- result out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const

Static Public Attributes

- static locale::id **id**
- static locale::id **id**

Protected Member Functions

- virtual bool do_always_noconv () const throw ()
- virtual bool do_always_noconv () const throw ()
- virtual int do_encoding () const throw ()
- virtual int do_encoding () const throw ()
- virtual result do_in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- virtual result do_in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- virtual int do_length (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const
- virtual int do_length (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const
- virtual int do_max_length () const throw ()
- virtual int do_max_length () const throw ()
- virtual result do_out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result do_out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result do_unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result do_unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale **_M_c_locale_codecvt**

5.281.1 Detailed Description

Class codecvt<char32_t, char, mbstate_t> specialization.
Converts between UTF-32 and UTF-8.

5.281.2 Member Function Documentation

do_always_noconv() [1/2]

```
virtual bool std::codecvt< char32_t, char, mbstate_t >::do_always_noconv () const throw ( )   [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< char32_t, char, mbstate_t >.

do_always_noconv() [2/2]

```
virtual bool std::codecvt< char32_t, char, mbstate_t >::do_always_noconv () const throw ( )   [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< char32_t, char, mbstate_t >.

do_encoding() [1/2]

```
virtual int std::codecvt< char32_t, char, mbstate_t >::do_encoding () const throw ( )   [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< char32_t, char, mbstate_t >.

do_encoding() [2/2]

```
virtual int std::codecvt< char32_t, char, mbstate_t >::do_encoding () const throw ( )   [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< char32_t, char, mbstate_t >.

do_in() [1/2]

```
virtual result std::codecvt< char32_t, char, mbstate_t >::do_in (
            state_type & __state,
            const extern_type * __from,
            const extern_type * __from_end,
            const extern_type *& __from_next,
            intern_type * __to,
            intern_type * __to_end,
            intern_type *& __to_next) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< char32_t, char, mbstate_t >.

do_in() [2/2]

```
virtual result std::codecvt< char32_t, char, mbstate_t >::do_in (
            state_type & __state,
            const extern_type * __from,
            const extern_type * __from_end,
            const extern_type *& __from_next,
            intern_type * __to,
            intern_type * __to_end,
            intern_type *& __to_next) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< char32_t, char, mbstate_t >.

do_length() [1/2]

```
virtual int std::codecvt< char32_t, char, mbstate_t >::do_length (
            state_type & ,
            const extern_type * __from,
            const extern_type * __end,
            size_t __max) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< char32_t, char, mbstate_t >.

do_length() [2/2]

```
virtual int std::codecvt< char32_t, char, mbstate_t >::do_length (
            state_type & ,
            const extern_type * __from,
            const extern_type * __end,
            size_t __max) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< char32_t, char, mbstate_t >.

do_max_length() [1/2]

```
virtual int std::codecvt< char32_t, char, mbstate_t >::do_max_length () const throw ( )   [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< char32_t, char, mbstate_t >.

do_max_length() [2/2]

```
virtual int std::codecvt< char32_t, char, mbstate_t >::do_max_length () const throw ( )   [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< char32_t, char, mbstate_t >.

do_out() [1/2]

```
virtual result std::codecvt< char32_t, char, mbstate_t >::do_out (
            state_type & __state,
            const intern_type * __from,
            const intern_type * __from_end,
            const intern_type *& __from_next,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [protected], [virtual]
```

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This function is a hook for derived classes to change the value returned.

See also

out for more information.

Implements std::__codecvt_abstract_base< char32_t, char, mbstate_t >.

do_out() [2/2]

```
virtual result std::codecvt< char32_t, char, mbstate_t >::do_out (
            state_type & __state,
            const intern_type * __from,
            const intern_type * __from_end,
            const intern_type *& __from_next,
```

```
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [protected], [virtual]
```

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This function is a hook for derived classes to change the value returned.

See also

out for more information.

Implements std::__codecvt_abstract_base< char32_t, char, mbstate_t >.

do_unshift() [1/2]

```
virtual result std::codecvt< char32_t, char, mbstate_t >::do_unshift (
            state_type & __state,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< char32_t, char, mbstate_t >.

do_unshift() [2/2]

```
virtual result std::codecvt< char32_t, char, mbstate_t >::do_unshift (
            state_type & __state,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< char32_t, char, mbstate_t >.

in() [1/2]

```
result std::__codecvt_abstract_base< char32_t, char, mbstate_t >::in (
            state_type & __state,
            const extern_type * __from,
            const extern_type * __from_end,
            const extern_type *& __from_next,
            intern_type * __to,
            intern_type * __to_end,
            intern_type *& __to_next) const  [inline], [inherited]
```

Convert from external to internal character set.

Converts input string of extern_type to output string of intern_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do_in.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.

Parameters

__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

in() [2/2]

```
result std::__codecvt_abstract_base< char32_t, char, mbstate_t >::in (
             state_type & __state,
             const extern_type * __from,
             const extern_type * __from_end,
             const extern_type *& __from_next,
             intern_type * __to,
             intern_type * __to_end,
             intern_type *& __to_next) const  [inline]
```

Convert from external to internal character set.

Converts input string of extern_type to output string of intern_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do_in.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

out() [1/2]

```
result std::__codecvt_abstract_base< char32_t, char, mbstate_t >::out (
             state_type & __state,
             const intern_type * __from,
             const intern_type * __from_end,
             const intern_type *& __from_next,
             extern_type * __to,
             extern_type * __to_end,
             extern_type *& __to_next) const  [inline], [inherited]
```

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This is analogous to wcsrtombs. It does this by calling codecvt::do_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

out() [2/2]

```
result std::__codecvt_abstract_base< char32_t, char, mbstate_t >::out (
             state_type & __state,
             const intern_type * __from,
             const intern_type * __from_end,
             const intern_type *& __from_next,
             extern_type * __to,
             extern_type * __to_end,
             extern_type *& __to_next) const  [inline]
```

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This is analogous to wcsrtombs. It does this by calling codecvt::do_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.
The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

unshift() [1/2]

```
result std::__codecvt_abstract_base< char32_t, char, mbstate_t >::unshift (
            state_type & __state,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [inline], [inherited]
```

Reset conversion state.
Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do_unshift().
For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.
The source and destination character sets are determined by the facet's locale, internal and external types.
The result returned is a member of codecvt_base::result. If the state could be reset and data written, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the output has insufficient space, returns codecvt_base::partial. Otherwise the reset failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

unshift() [2/2]

```
result std::__codecvt_abstract_base< char32_t, char, mbstate_t >::unshift (
            state_type & __state,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [inline]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of codecvt_base::result. If the state could be reset and data written, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the output has insufficient space, returns codecvt_base::partial. Otherwise the reset failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

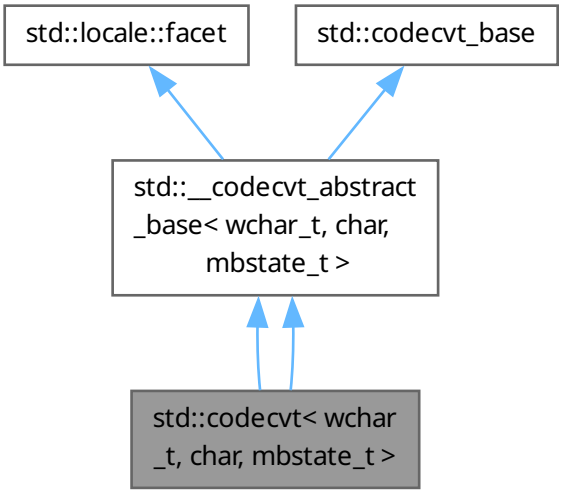
The documentation for this class was generated from the following file:

- codecvt.h

5.282 std::codecvt< wchar_t, char, mbstate_t > Class Reference

```
#include <codecvt.h>
```

Inheritance diagram for std::codecvt< wchar_t, char, mbstate_t >:



Public Types

- typedef char **extern_type**
- typedef char **extern_type**
- typedef wchar_t **intern_type**
- typedef wchar_t **intern_type**
- typedef codecvt_base::result **result**
- typedef mbstate_t **state_type**
- typedef mbstate_t **state_type**

Public Member Functions

- **codecvt** (__c_locale __cloc, size_t __refs=0)
- **codecvt** (__c_locale __cloc, size_t __refs=0)
- **codecvt** (size_t __refs=0)
- **codecvt** (size_t __refs=0)
- bool **always_noconv** () const throw ()
- bool **always_noconv** () const throw ()
- int **encoding** () const throw ()
- int **encoding** () const throw ()
- result in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- result in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- int **length** (state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max) const
- int **length** (state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max) const
- int **max_length** () const throw ()

- int **max_length** () const throw ()
- result out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const

Static Public Attributes

- static locale::id **id**
- static locale::id **id**

Protected Member Functions

- virtual bool do_always_noconv () const throw ()
- virtual bool do_always_noconv () const throw ()
- virtual int do_encoding () const throw ()
- virtual int do_encoding () const throw ()
- virtual result do_in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- virtual result do_in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- virtual int do_length (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const
- virtual int do_length (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const
- virtual int do_max_length () const throw ()
- virtual int do_max_length () const throw ()
- virtual result do_out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result do_out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result do_unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result do_unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale **_M_c_locale_codecvt**
- __c_locale **_M_c_locale_codecvt**

Friends

- class **messages**< **wchar_t** >

5.282.1 Detailed Description

Class codecvt<wchar_t, char, mbstate_t> specialization.
Converts between narrow and wide characters in the native character set

5.282.2 Member Function Documentation

do_always_noconv() [1/2]

```
virtual bool std::codecvt< wchar_t, char, mbstate_t >::do_always_noconv () const throw ( )   [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< wchar_t, char, mbstate_t >.

do_always_noconv() [2/2]

```
virtual bool std::codecvt< wchar_t, char, mbstate_t >::do_always_noconv () const throw ( )   [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< wchar_t, char, mbstate_t >.

do_encoding() [1/2]

```
virtual int std::codecvt< wchar_t, char, mbstate_t >::do_encoding () const throw ( )   [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< wchar_t, char, mbstate_t >.

do_encoding() [2/2]

```
virtual int std::codecvt< wchar_t, char, mbstate_t >::do_encoding () const throw ( )   [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< wchar_t, char, mbstate_t >.

do_in() [1/2]

```
virtual result std::codecvt< wchar_t, char, mbstate_t >::do_in (
            state_type & __state,
            const extern_type * __from,
            const extern_type * __from_end,
            const extern_type *& __from_next,
            intern_type * __to,
            intern_type * __to_end,
            intern_type *& __to_next) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< wchar_t, char, mbstate_t >.

do_in() [2/2]

```
virtual result std::codecvt< wchar_t, char, mbstate_t >::do_in (
            state_type & __state,
```

```
            const extern_type * __from,
            const extern_type * __from_end,
            const extern_type *& __from_next,
            intern_type * __to,
            intern_type * __to_end,
            intern_type *& __to_next) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< wchar_t, char, mbstate_t >.

do_length() [1/2]

```
virtual int std::codecvt< wchar_t, char, mbstate_t >::do_length (
            state_type & ,
            const extern_type * __from,
            const extern_type * __end,
            size_t __max) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< wchar_t, char, mbstate_t >.

do_length() [2/2]

```
virtual int std::codecvt< wchar_t, char, mbstate_t >::do_length (
            state_type & ,
            const extern_type * __from,
            const extern_type * __end,
            size_t __max) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< wchar_t, char, mbstate_t >.

do_max_length() [1/2]

```
virtual int std::codecvt< wchar_t, char, mbstate_t >::do_max_length () const throw ( )   [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< wchar_t, char, mbstate_t >.

do_max_length() [2/2]

```
virtual int std::codecvt< wchar_t, char, mbstate_t >::do_max_length () const throw ( )   [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< wchar_t, char, mbstate_t >.

do_out() [1/2]

```
virtual result std::codecvt< wchar_t, char, mbstate_t >::do_out (
            state_type & __state,
            const intern_type * __from,
            const intern_type * __from_end,
            const intern_type *& __from_next,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [protected], [virtual]
```

Convert from internal to external character set.
Converts input string of intern_type to output string of extern_type. This function is a hook for derived classes to change the value returned.

See also

out for more information.

Implements std::__codecvt_abstract_base< wchar_t, char, mbstate_t >.

do_out() [2/2]

```
virtual result std::codecvt< wchar_t, char, mbstate_t >::do_out (
            state_type & __state,
            const intern_type * __from,
            const intern_type * __from_end,
            const intern_type *& __from_next,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [protected], [virtual]
```

Convert from internal to external character set.
Converts input string of intern_type to output string of extern_type. This function is a hook for derived classes to change the value returned.

See also

out for more information.

Implements std::__codecvt_abstract_base< wchar_t, char, mbstate_t >.

do_unshift() [1/2]

```
virtual result std::codecvt< wchar_t, char, mbstate_t >::do_unshift (
            state_type & __state,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< wchar_t, char, mbstate_t >.

do_unshift() [2/2]

```
virtual result std::codecvt< wchar_t, char, mbstate_t >::do_unshift (
            state_type & __state,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< wchar_t, char, mbstate_t >.

in() [1/2]

```
result std::__codecvt_abstract_base< wchar_t, char, mbstate_t >::in (
            state_type & __state,
            const extern_type * __from,
            const extern_type * __from_end,
            const extern_type *& __from_next,
            intern_type * __to,
            intern_type * __to_end,
            intern_type *& __to_next) const  [inline], [inherited]
```

Convert from external to internal character set.
Converts input string of extern_type to output string of intern_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do_in.
The source and destination character sets are determined by the facet's locale, internal and external types.
The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.
The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

in() [2/2]

```
result std::__codecvt_abstract_base< wchar_t, char, mbstate_t >::in (
             state_type & __state,
             const extern_type * __from,
             const extern_type * __from_end,
             const extern_type *& __from_next,
             intern_type * __to,
             intern_type * __to_end,
             intern_type *& __to_next) const  [inline]
```

Convert from external to internal character set.

Converts input string of extern_type to output string of intern_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do_in.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

out() [1/2]

```
result std::__codecvt_abstract_base< wchar_t, char, mbstate_t >::out (
             state_type & __state,
             const intern_type * __from,
             const intern_type * __from_end,
             const intern_type *& __from_next,
             extern_type * __to,
             extern_type * __to_end,
             extern_type *& __to_next) const  [inline], [inherited]
```

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This is analogous to wcsrtombs. It does this by calling codecvt::do_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

out() [2/2]

```
result std::__codecvt_abstract_base< wchar_t, char, mbstate_t >::out (
             state_type & __state,
             const intern_type * __from,
             const intern_type * __from_end,
             const intern_type *& __from_next,
             extern_type * __to,
             extern_type * __to_end,
             extern_type *& __to_next) const  [inline]
```

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This is analogous to wcsrtombs. It does this by calling codecvt::do_out.
The source and destination character sets are determined by the facet's locale, internal and external types.
The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.
The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.
The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

unshift() [1/2]

```
result std::__codecvt_abstract_base< wchar_t, char, mbstate_t >::unshift (
            state_type & __state,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [inline], [inherited]
```

Reset conversion state.
Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do_unshift().
For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.
The source and destination character sets are determined by the facet's locale, internal and external types.
The result returned is a member of codecvt_base::result. If the state could be reset and data written, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the output has insufficient space, returns codecvt_base::partial. Otherwise the reset failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

unshift() [2/2]

```
result std::__codecvt_abstract_base< wchar_t, char, mbstate_t >::unshift (
            state_type & __state,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [inline]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of codecvt_base::result. If the state could be reset and data written, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the output has insufficient space, returns codecvt_base::partial. Otherwise the reset failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

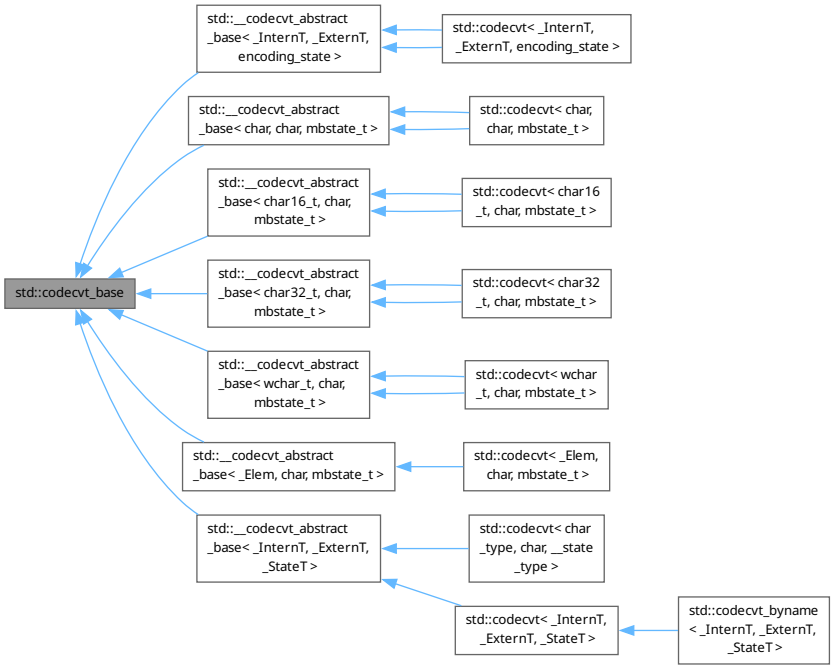
The documentation for this class was generated from the following file:

- codecvt.h

5.283 std::codecvt_base Class Reference

```
#include <codecvt.h>
```

Inheritance diagram for std::codecvt_base:



Public Types

- enum **result** { **ok** , **partial** , **error** , **noconv** }

5.283.1 Detailed Description

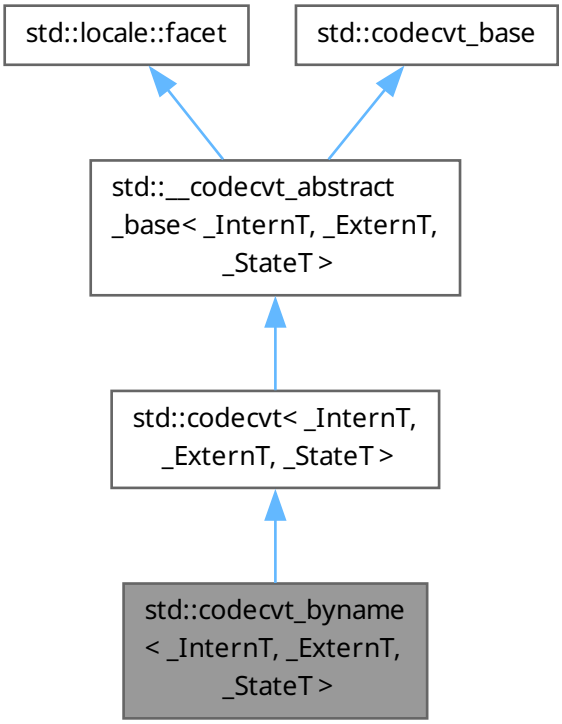
Empty base class for codecvt facet [22.2.1.5].
The documentation for this class was generated from the following file:

- codecvt.h

5.284 std::codecvt_byname< _InternT, _ExternT, _StateT > Class Template Reference

`#include <codecvt.h>`

Inheritance diagram for std::codecvt_byname< _InternT, _ExternT, _StateT >:



Public Types

- typedef _ExternT **extern_type**
- typedef _InternT **intern_type**
- typedef codecvt_base::result **result**
- typedef _StateT **state_type**

Public Member Functions

- **codecvt_byname** (const char *__s, size_t __refs=0)
- **codecvt_byname** (const string &__s, size_t __refs=0)
- bool **always_noconv** () const throw ()
- int **encoding** () const throw ()
- result in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- int **length** (state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max) const
- int **max_length** () const throw ()
- result out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const

Static Public Attributes

- static locale::id **id**

Protected Member Functions

- virtual bool do_always_noconv () const throw ()
- virtual int do_encoding () const throw ()
- virtual result do_in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- virtual int do_length (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const
- virtual int do_max_length () const throw ()
- virtual result do_out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result do_unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to↩ _next) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale **_M_c_locale_codecvt**

5.284.1 Detailed Description

template<typename _InternT, typename _ExternT, typename _StateT>
class std::codecvt_byname< _InternT, _ExternT, _StateT >

class codecvt_byname [22.2.1.6].

5.284.2 Member Function Documentation

do_always_noconv()

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual bool std::codecvt< _InternT, _ExternT, _StateT >::do_always_noconv () const throw ( )
[protected], [virtual], [inherited]
```

Implements std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >.

do_encoding()

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual int std::codecvt< _InternT, _ExternT, _StateT >::do_encoding () const throw ( )   [protected],
[virtual], [inherited]
```

Implements std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >.

do_in()

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual result std::codecvt< _InternT, _ExternT, _StateT >::do_in (
            state_type & __state,
            const extern_type * __from,
            const extern_type * __from_end,
            const extern_type *& __from_next,
            intern_type * __to,
            intern_type * __to_end,
            intern_type *& __to_next) const  [protected], [virtual], [inherited]
```

Implements std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >.

do_length()

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual int std::codecvt< _InternT, _ExternT, _StateT >::do_length (
            state_type & ,
            const extern_type * __from,
            const extern_type * __end,
            size_t __max) const  [protected], [virtual], [inherited]
```

Implements std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >.

do_max_length()

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual int std::codecvt< _InternT, _ExternT, _StateT >::do_max_length () const throw ( )   [protected],
[virtual], [inherited]
```

Implements std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >.

do_out()

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual result std::codecvt< _InternT, _ExternT, _StateT >::do_out (
            state_type & __state,
            const intern_type * __from,
            const intern_type * __from_end,
            const intern_type *& __from_next,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [protected], [virtual], [inherited]
```

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This function is a hook for derived classes to change the value returned.

See also

out for more information.

Implements std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >.

do_unshift()

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual result std::codecvt< _InternT, _ExternT, _StateT >::do_unshift (
            state_type & __state,
            extern_type * __to,
```

```
            extern_type * __to_end,
            extern_type *& __to_next) const  [protected], [virtual], [inherited]
```

Implements std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >.

in()

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::in (
            state_type & __state,
            const extern_type * __from,
            const extern_type * __from_end,
            const extern_type *& __from_next,
            intern_type * __to,
            intern_type * __to_end,
            intern_type *& __to_next) const  [inline], [inherited]
```

Convert from external to internal character set.

Converts input string of extern_type to output string of intern_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do_in.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

out()

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::out (
            state_type & __state,
            const intern_type * __from,
            const intern_type * __from_end,
            const intern_type *& __from_next,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [inline], [inherited]
```

Convert from internal to external character set.
Converts input string of intern_type to output string of extern_type. This is analogous to wcsrtombs. It does this by calling codecvt::do_out.
The source and destination character sets are determined by the facet's locale, internal and external types.
The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.
The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.
The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

References do_out().

unshift()

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::unshift (
             state_type & __state,
             extern_type * __to,
             extern_type * __to_end,
             extern_type *& __to_next) const  [inline], [inherited]
```

Reset conversion state.
Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do_unshift().
For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.
The source and destination character sets are determined by the facet's locale, internal and external types.
The result returned is a member of codecvt_base::result. If the state could be reset and data written, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the output has insufficient space, returns codecvt_base::partial. Otherwise the reset failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__to	Start of output buffer.
__to_end	End of output buffer.

__to_next	Returns start of unused output area.

Returns

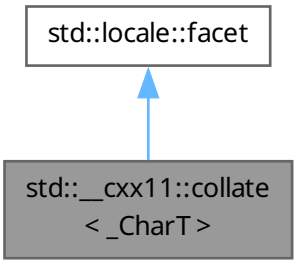
codecvt_base::result.

The documentation for this class was generated from the following file:

- codecvt.h

5.285 std::__cxx11::collate< _CharT > Class Template Reference

`#include <locale_classes.h>`
Inheritance diagram for std::__cxx11::collate< _CharT >:



Public Types

- typedef _CharT char_type
- typedef basic_string< _CharT > string_type

Public Member Functions

- collate (__c_locale __cloc, size_t __refs=0)
- collate (size_t __refs=0)
- int **_M_compare** (const _CharT *, const _CharT *) const throw ()
- size_t **_M_transform** (_CharT *, const _CharT *, size_t) const throw ()
- int compare (const _CharT *__lo1, const _CharT *__hi1, const _CharT *__lo2, const _CharT *__hi2) const
- long hash (const _CharT *__lo, const _CharT *__hi) const
- string_type transform (const _CharT *__lo, const _CharT *__hi) const

Static Public Attributes

- static locale::id id

Protected Member Functions

- virtual ~collate ()
- virtual int do_compare (const _CharT *__lo1, const _CharT *__hi1, const _CharT *__lo2, const _CharT *__hi2) const
- virtual long do_hash (const _CharT *__lo, const _CharT *__hi) const
- virtual string_type do_transform (const _CharT *__lo, const _CharT *__hi) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale **_M_c_locale_collate**

5.285.1 Detailed Description

template<typename _CharT>
class std::__cxx11::collate< _CharT >

Facet for localized string comparison.
This facet encapsulates the code to compare strings in a localized manner.
The collate template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the collate facet.

5.285.2 Member Typedef Documentation

char_type

```
template<typename _CharT>
typedef _CharT std::__cxx11::collate< _CharT >::char_type
```

Public typedefs.

string_type

```
template<typename _CharT>
typedef basic_string<_CharT> std::__cxx11::collate< _CharT >::string_type
```

Public typedefs.

5.285.3 Constructor & Destructor Documentation

collate() [1/2]

```
template<typename _CharT>
std::__cxx11::collate< _CharT >::collate (
            size_t __refs = 0)  [inline], [explicit]
```

Constructor performs initialization.
This is the constructor provided by the standard.

Parameters

__refs	Passed to the base facet class.

References std::locale::facet::facet().
Referenced by do_compare(), do_hash(), and do_transform().

collate() [2/2]

```
template<typename _CharT>
std::__cxx11::collate< _CharT >::collate (
            __c_locale __cloc,
            size_t __refs = 0)  [inline], [explicit]
```

Internal constructor. Not for general use.
This is a constructor for use by the library itself to set up new locales.

Parameters

__cloc	The C locale.
__refs	Passed to the base facet class.

References std::locale::facet::facet().

~collate()

```
template<typename _CharT>
virtual std::__cxx11::collate< _CharT >::~collate ()  [inline], [protected], [virtual]
```

Destructor.

5.285.4 Member Function Documentation

compare()

```
template<typename _CharT>
int std::__cxx11::collate< _CharT >::compare (
            const _CharT * __lo1,
            const _CharT * __hi1,
            const _CharT * __lo2,
            const _CharT * __hi2) const  [inline]
```

Compare two strings.
This function compares two strings and returns the result by calling collate::do_compare().

Parameters

__lo1	Start of string 1.
__hi1	End of string 1.
__lo2	Start of string 2.
__hi2	End of string 2.

Returns

1 if string1 > string2, -1 if string1 < string2, else 0.

References do_compare().

do_compare()

```
template<typename _CharT>
int std::collate< _CharT >::do_compare (
            const _CharT * __lo1,
            const _CharT * __hi1,
            const _CharT * __lo2,
            const _CharT * __hi2) const  [protected], [virtual]
```

Compare two strings.
This function is a hook for derived classes to change the value returned.

See also

compare().

Parameters

__lo1	Start of string 1.
__hi1	End of string 1.
__lo2	Start of string 2.
__hi2	End of string 2.

Returns

1 if string1 > string2, -1 if string1 < string2, else 0.

References collate(), std::basic_string< _CharT, _Traits, _Alloc >::c_str(), std::basic_string< _CharT, _Traits, _Alloc >::data(), and std::basic_string< _CharT, _Traits, _Alloc >::length().
Referenced by compare().

do_hash()

```
template<typename _CharT>
long std::collate< _CharT >::do_hash (
            const _CharT * __lo,
            const _CharT * __hi) const  [protected], [virtual]
```

Return hash of a string.
This function computes and returns a hash on the input string. This function is a hook for derived classes to change the value returned.

Parameters

__lo	Start of string.
__hi	End of string.

Returns

Hash value.

References collate().
Referenced by hash().

do_transform()

```
template<typename _CharT>
collate< _CharT >::string_type std::collate< _CharT >::do_transform (
            const _CharT * __lo,
            const _CharT * __hi) const  [protected], [virtual]
```

Transform string to comparable form.
This function is a hook for derived classes to change the value returned.

Parameters

_↩ _lo	Start.
_↩ _hi	End.

Returns

transformed string.

References collate(), std::basic_string< _CharT, _Traits, _Alloc >::append(), std::basic_string< _CharT, _Traits, _Alloc >::c_str(), std::basic_string< _CharT, _Traits, _Alloc >::clear(), std::basic_string< _CharT, _Traits, _Alloc >::data(), std::basic_string< _CharT, _Tra and std::basic_string< _CharT, _Traits, _Alloc >::push_back().
Referenced by transform().

hash()

```
template<typename _CharT>
long std::__cxx11::collate< _CharT >::hash (
            const _CharT * __lo,
            const _CharT * __hi) const  [inline]
```

Return hash of a string.
This function computes and returns a hash on the input string. It does so by returning collate::do_hash().

Parameters

_↩ _lo	Start of string.
_↩ _hi	End of string.

Returns

Hash value.

References do_hash().

transform()

```
template<typename _CharT>
string_type std::__cxx11::collate< _CharT >::transform (
            const _CharT * __lo,
            const _CharT * __hi) const  [inline]
```

Transform string to comparable form.

This function is a wrapper for strxfrm functionality. It takes the input string and returns a modified string that can be directly compared to other transformed strings. In the C locale, this function just returns a copy of the input string. In some other locales, it may replace two chars with one, change a char for another, etc. It does so by returning collate::do_transform().

Parameters

_↩ _lo	Start of string.
_↩ _hi	End of string.

Returns

Transformed string_type.

References do_transform().

5.285.5 Member Data Documentation

id

```
template<typename _CharT>
locale::id std::__cxx11::collate< _CharT >::id  [static]
```

Numpunct facet id.

The documentation for this class was generated from the following files:

- locale_classes.h
- locale_classes.tcc

5.286 std::__cxx11::collate_byname< _CharT > Class Template Reference

```
#include <locale_classes.h>
```

Public Types

- typedef _CharT char_type
- typedef basic_string< _CharT > string_type

Public Member Functions

- **collate_byname** (const char *__s, size_t __refs=0)
- **collate_byname** (const string &__s, size_t __refs=0)
- int **_M_compare** (const char *, const char *) const throw()
- int **_M_compare** (const wchar_t *, const wchar_t *) const throw()
- size_t **_M_transform** (char *, const char *, size_t) const throw()
- size_t **_M_transform** (wchar_t *, const wchar_t *, size_t) const throw()

5.286.1 Detailed Description

template<typename _CharT>
class std::__cxx11::collate_byname< _CharT >

class collate_byname [22.2.4.2].

5.286.2 Member Typedef Documentation

char_type

```
template<typename _CharT>
typedef _CharT std::__cxx11::collate_byname< _CharT >::char_type
```

Public typedefs.

string_type

```
template<typename _CharT>
typedef basic_string<_CharT> std::__cxx11::collate_byname< _CharT >::string_type
```

Public typedefs.

The documentation for this class was generated from the following file:

- locale_classes.h

5.287 std::common_iterator< _It, _Sent > Class Template Reference

```
#include <stl_iterator.h>
```

Public Member Functions

- constexpr **common_iterator** (_It __i) noexcept(is_nothrow_move_constructible_v< _It >)
- constexpr **common_iterator** (_Sent __s) noexcept(is_nothrow_move_constructible_v< _Sent >)
- **common_iterator** (common_iterator &&)=default
- constexpr **common_iterator** (common_iterator &&__x) noexcept(_S_noexcept< _It, _Sent >())
- **common_iterator** (const common_iterator &)=default
- constexpr **common_iterator** (const common_iterator &__x) noexcept(_S_noexcept< const _It &, const _Sent & >())
- template<typename _It2, typename _Sent2>
  requires convertible_to<const _It2&, _It> && convertible_to<const _Sent2&, _Sent>
  constexpr **common_iterator** (const common_iterator< _It2, _Sent2 > &__x) noexcept(_S_noexcept< const _It2 &, const _Sent2 & >())
- constexpr decltype(auto) **operator*** ()
- constexpr decltype(auto) **operator*** () const
- constexpr common_iterator & **operator++** ()
- constexpr decltype(auto) **operator++** (int)
- constexpr auto **operator->** () const
- constexpr common_iterator & **operator=** (common_iterator &&)=default
- constexpr common_iterator & **operator=** (common_iterator &&__x) noexcept(is_nothrow_move_assignable_v< _It > &&is_nothrow_move_assignable_v< _Sent > &&is_nothrow_move_constructible_v< _It > &&is_nothrow_move_constructible_v< _Sent >)
- constexpr common_iterator & **operator=** (const common_iterator &)=default
- constexpr common_iterator & **operator=** (const common_iterator &__x) noexcept(is_nothrow_copy_assignable_v< _It > &&is_nothrow_copy_assignable_v< _Sent > &&is_nothrow_copy_constructible_v< _It > &&is_nothrow_copy_constructible_v< _Sent >)
- template<typename _It2, typename _Sent2>
  requires convertible_to<const _It2&, _It> && convertible_to<const _Sent2&, _Sent> && assignable_from<_It&, const _It2&> && assignable_from<_Sent&, const _Sent2&>
  constexpr common_iterator & **operator=** (const common_iterator< _It2, _Sent2 > &__x) noexcept(is_nothrow_constructible_v< _It, const _It2 & > &&is_nothrow_constructible_v< _Sent, const _Sent2 & > &&is_nothrow_assignable_v< _It &, const _It2 & > &&is_nothrow_assignable_v< _Sent &, const _Sent2 & >)

Friends

- constexpr iter_rvalue_reference_t< _It > **iter_move** (const common_iterator &__i) noexcept(noexcept(ranges::iter_move(std::declval< const _It & >())))
- template<indirectly_swappable< _It > _It2, typename _Sent2>
  constexpr void **iter_swap** (const common_iterator &__x, const common_iterator< _It2, _Sent2 > &__y) noexcept(noexcept(ranges::iter_swap(std::declval< const _It & >(), std::declval< const _It2 & >())))
- template<sized_sentinel_for< _It > _It2, sized_sentinel_for< _It > _Sent2>
  requires sized_sentinel_for<_Sent, _It2>
  constexpr iter_difference_t< _It2 > **operator-** (const common_iterator &__x, const common_iterator< _It2, _Sent2 > &__y)
- template<typename _It2, sentinel_for< _It > _Sent2>
  requires sentinel_for<_Sent, _It2>
  constexpr bool **operator==** (const common_iterator &__x, const common_iterator< _It2, _Sent2 > &__y)
- template<typename _It2, sentinel_for< _It > _Sent2>
  requires sentinel_for<_Sent, _It2> && equality_comparable_with<_It, _It2>
  constexpr bool **operator==** (const common_iterator &__x, const common_iterator< _It2, _Sent2 > &__y)

5.287.1 Detailed Description

template<input_or_output_iterator _It, sentinel_for< _It > _Sent>
requires (!same_as<_It, _Sent>) && copyable<_It>
class std::common_iterator< _It, _Sent >

An iterator/sentinel adaptor for representing a non-common range.
The documentation for this class was generated from the following file:

- bits/stl_iterator.h

5.288 std::common_type< _Tp > Struct Template Reference

5.288.1 Detailed Description

template<typename... _Tp>
struct std::common_type< _Tp >

common_type
The documentation for this struct was generated from the following file:

- type_traits

5.289 std::common_type< chrono::duration< _Rep, _Period > > Struct Template Reference

`#include <chrono.h>`

Public Types

- using **type**

5.289.1 Detailed Description

template<typename _Rep, typename _Period>
struct std::common_type< chrono::duration< _Rep, _Period > >

Specialization of common_type for one chrono::duration type.
The documentation for this struct was generated from the following file:

- chrono.h

5.290 std::common_type< chrono::duration< _Rep, _Period >, chrono::duration< _Rep, _Period > > Struct Template Reference

`#include <chrono.h>`

Public Types

- using **type**

5.290.1 Detailed Description

template<typename _Rep, typename _Period>
struct std::common_type< chrono::duration< _Rep, _Period >, chrono::duration< _Rep, _Period > >

Specialization of common_type for two identical chrono::duration types.
The documentation for this struct was generated from the following file:

- chrono.h

5.291 std::common_type< chrono::duration< _Rep1, _Period1 >, chrono::duration< _Rep2, _Period2 > > Struct Template Reference

`#include <chrono.h>`

5.291.1 Detailed Description

template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
struct std::common_type< chrono::duration< _Rep1, _Period1 >, chrono::duration< _Rep2, _Period2 > >

Specialization of common_type for chrono::duration types.
The documentation for this struct was generated from the following file:

- chrono.h

5.292 std::common_type< chrono::time_point< _Clock, _Duration > > Struct Template Reference

`#include <chrono.h>`

Public Types

- using **type**

5.292.1 Detailed Description

template<typename _Clock, typename _Duration>
struct std::common_type< chrono::time_point< _Clock, _Duration > >

Specialization of common_type for one chrono::time_point type.
The documentation for this struct was generated from the following file:

- chrono.h

5.293 std::common_type< chrono::time_point< _Clock, _Duration >, chrono::time_point< _Clock, _Duration > > Struct Template Reference

`#include <chrono.h>`

Public Types

- using **type**

5.293.1 Detailed Description

template<typename _Clock, typename _Duration>
struct std::common_type< chrono::time_point< _Clock, _Duration >, chrono::time_point< _Clock, _Duration > >

Specialization of common_type for two identical chrono::time_point types.
The documentation for this struct was generated from the following file:

- chrono.h

5.294 std::common_type< chrono::time_point< _Clock, _Duration1 >, chrono::time_point< _Clock, _Duration2 > > Struct Template Reference

`#include <chrono.h>`

5.294.1 Detailed Description

template<typename _Clock, typename _Duration1, typename _Duration2>
struct std::common_type< chrono::time_point< _Clock, _Duration1 >, chrono::time_point< _Clock, _Duration2 > >

Specialization of common_type for chrono::time_point types.
The documentation for this struct was generated from the following file:

- chrono.h

5.295 std::compare_three_way_result< _Tp, _Up > Struct Template Reference

`#include <compare>`

5.295.1 Detailed Description

template<typename _Tp, typename _Up = _Tp>
struct std::compare_three_way_result< _Tp, _Up >

[cmp.result], result of three-way comparison
The documentation for this struct was generated from the following file:

- compare

5.296 std::complex< _Tp > Class Template Reference

`#include <complex>`

Public Types

- typedef _Tp value_type

Public Member Functions

- constexpr complex (const _Tp &__r=_Tp(), const _Tp &__i=_Tp())
- constexpr **complex** (const complex &)=default
- template<typename _Up>
  constexpr complex (const complex< _Up > &__z)
- **__attribute** ((__abi_tag__("cxx11"))) const expr _Tp imag() const
- **__attribute** ((__abi_tag__("cxx11"))) const expr _Tp real() const
- constexpr complex **__rep** () const
- constexpr void **imag** (_Tp __val)
- constexpr complex< _Tp > & operator*= (const _Tp &)
- template<typename _Up>
  constexpr complex< _Tp > & operator*= (const complex< _Up > &)
- constexpr complex< _Tp > & operator+= (const _Tp &__t)
- template<typename _Up>
  constexpr complex< _Tp > & operator+= (const complex< _Up > &)
- constexpr complex< _Tp > & operator-= (const _Tp &__t)
- template<typename _Up>
  constexpr complex< _Tp > & operator-= (const complex< _Up > &)
- constexpr complex< _Tp > & operator/= (const _Tp &)
- template<typename _Up>
  constexpr complex< _Tp > & operator/= (const complex< _Up > &)
- constexpr complex< _Tp > & operator= (const _Tp &)
- constexpr complex & **operator=** (const complex &)=default
- template<typename _Up>
  constexpr complex< _Tp > & operator= (const complex< _Up > &)
- constexpr void **real** (_Tp __val)

5.296.1 Detailed Description

template<typename _Tp>
class std::complex< _Tp >

Template to represent complex numbers.
Specializations for float, double, and long double are part of the library. Results with any other type are not guaranteed.

Parameters

Tp	Type of real and imaginary values.

5.296.2 Member Typedef Documentation

value_type

```
template<typename _Tp>
typedef _Tp std::complex< _Tp >::value_type
```

Value typedef.

5.296.3 Constructor & Destructor Documentation

complex() [1/2]

```
template<typename _Tp>
std::complex< _Tp >::complex (
            const _Tp & __r = _Tp(),
            const _Tp & __i = _Tp())  [inline], [constexpr]
```

Default constructor. First parameter is x, second parameter is y. Unspecified parameters default to 0.

complex() [2/2]

```
template<typename _Tp>
template<typename _Up>
std::complex< _Tp >::complex (
            const complex< _Up > & __z)  [inline], [constexpr]
```

Converting constructor.

5.296.4 Member Function Documentation

operator+=()

```
template<typename _Tp>
complex< _Tp > & std::complex< _Tp >::operator+= (
            const _Tp & __t)  [inline], [constexpr]
```

Add a scalar to this complex number.

operator-=()

```
template<typename _Tp>
complex< _Tp > & std::complex< _Tp >::operator-= (
            const _Tp & __t)  [inline], [constexpr]
```

Subtract a scalar from this complex number.
The documentation for this class was generated from the following file:

- complex

5.297 std::complex< double > Class Reference

`#include <complex>`

Public Types

- typedef __complex__ double **_ComplexT**
- typedef double value_type
- typedef double **value_type**

Public Member Functions

- constexpr **complex** (_ComplexT __z)
- constexpr **complex** (const complex &)=default
- constexpr **complex** (const complex &)=default
- constexpr complex (const complex< _Up > &__z)
- constexpr **complex** (const complex< float > &__z)
- constexpr **complex** (const complex< long double > &)
- constexpr complex (const double &__r=double(), const double &__i=double())

- constexpr **complex** (double __r=0.0, double __i=0.0)
- **__attribute** ((__abi_tag__("cxx11"))) const expr double imag() const
- **__attribute** ((__abi_tag__("cxx11"))) const expr double imag() const
- **__attribute** ((__abi_tag__("cxx11"))) const expr double real() const
- **__attribute** ((__abi_tag__("cxx11"))) const expr double real() const
- constexpr complex **__rep** () const
- constexpr _ComplexT **__rep** () const
- constexpr void **imag** (double __val)
- constexpr void **imag** (double __val)
- template<typename _Tp>
  constexpr complex & **operator*=** (const complex< _Tp > &__z)
- constexpr complex< double > & operator*= (const complex< _Up > &)
- constexpr complex< double > & operator*= (const double &)
- constexpr complex & **operator*=** (double __d)
- template<typename _Tp>
  constexpr complex & **operator+=** (const complex< _Tp > &__z)
- constexpr complex< double > & operator+= (const complex< _Up > &)
- constexpr complex< double > & operator+= (const double &__t)
- constexpr complex & **operator+=** (double __d)
- template<typename _Tp>
  constexpr complex & **operator-=** (const complex< _Tp > &__z)
- constexpr complex< double > & operator-= (const complex< _Up > &)
- constexpr complex< double > & operator-= (const double &__t)
- constexpr complex & **operator-=** (double __d)
- template<typename _Tp>
  constexpr complex & **operator/=** (const complex< _Tp > &__z)
- constexpr complex< double > & operator/= (const complex< _Up > &)
- constexpr complex< double > & operator/= (const double &)
- constexpr complex & **operator/=** (double __d)
- constexpr complex & **operator=** (const complex &)=default
- constexpr complex & **operator=** (const complex &)=default
- template<typename _Tp>
  constexpr complex & **operator=** (const complex< _Tp > &__z)
- constexpr complex< double > & operator= (const complex< _Up > &)
- constexpr complex< double > & operator= (const double &)
- constexpr complex & **operator=** (double __d)
- constexpr void **real** (double __val)
- constexpr void **real** (double __val)

5.297.1 Detailed Description

26.2.3 complex specializations complex<double> specialization

5.297.2 Member Typedef Documentation

value_type

```
typedef double std::complex< double >::value_type
```

Value typedef.

5.297.3 Constructor & Destructor Documentation

complex() [1/2]

```
std::complex< double >::complex (
            const double & __r = _Tp(),
            const double & __i = _Tp())  [inline], [constexpr]
```

Default constructor. First parameter is x, second parameter is y. Unspecified parameters default to 0.

complex() [2/2]

```
std::complex< double >::complex (
            const complex< _Up > & __z)  [inline], [constexpr]
```

Converting constructor.

5.297.4 Member Function Documentation

operator*=() [1/2]

```
complex< double > & std::complex< double >::operator*= (
            const complex< _Up > & __z)  [constexpr]
```

Multiply this complex number by another.

operator*=() [2/2]

```
complex< double > & std::complex< double >::operator*= (
            const double & __t)  [constexpr]
```

Multiply this complex number by a scalar.

operator+=() [1/2]

```
complex< double > & std::complex< double >::operator+= (
            const complex< _Up > & __z)  [constexpr]
```

Add another complex number to this one.

operator+=() [2/2]

```
complex< double > & std::complex< double >::operator+= (
            const double & __t)  [inline], [constexpr]
```

Add a scalar to this complex number.

operator-=() [1/2]

```
complex< double > & std::complex< double >::operator-= (
            const complex< _Up > & __z)  [constexpr]
```

Subtract another complex number from this one.

operator-=() [2/2]

```
complex< double > & std::complex< double >::operator-= (
            const double & __t)  [inline], [constexpr]
```

Subtract a scalar from this complex number.

operator/=() [1/2]

```
complex< double > & std::complex< double >::operator/= (
            const complex< _Up > & __z)  [constexpr]
```

Divide this complex number by another.

operator/=() [2/2]

```
complex< double > & std::complex< double >::operator/= (
            const double & __t)  [constexpr]
```

Divide this complex number by a scalar.

operator=() [1/2]

```
complex< double > & std::complex< double >::operator= (
            const complex< _Up > & __z)  [constexpr]
```

Assign another complex number to this one.

operator=() [2/2]

```
complex< double > & std::complex< double >::operator= (
            const double & __t)  [constexpr]
```

Assign a scalar to this complex number.
The documentation for this class was generated from the following file:

- complex

5.298 std::complex< float > Class Reference

`#include <complex>`

Public Types

- typedef __complex__ float **_ComplexT**
- typedef float value_type
- typedef float **value_type**

Public Member Functions

- constexpr **complex** (_ComplexT __z)
- constexpr **complex** (const complex &)=default
- constexpr **complex** (const complex &)=default
- constexpr complex (const complex< _Up > &__z)
- constexpr **complex** (const complex< double > &)
- constexpr **complex** (const complex< long double > &)
- constexpr complex (const float &__r=float(), const float &__i=float())
- constexpr **complex** (float __r=0.0f, float __i=0.0f)
- **__attribute** ((__abi_tag__("cxx11"))) const expr float imag() const
- **__attribute** ((__abi_tag__("cxx11"))) const expr float imag() const
- **__attribute** ((__abi_tag__("cxx11"))) const expr float real() const
- **__attribute** ((__abi_tag__("cxx11"))) const expr float real() const
- constexpr complex **__rep** () const
- constexpr _ComplexT **__rep** () const
- constexpr void **imag** (float __val)

- constexpr void **imag** (float __val)
- template<class _Tp>
  constexpr complex & **operator*=** (const complex< _Tp > &__z)
- constexpr complex< float > & operator*= (const complex< _Up > &)
- constexpr complex< float > & operator*= (const float &)
- constexpr complex & **operator*=** (float __f)
- template<typename _Tp>
  constexpr complex & **operator+=** (const complex< _Tp > &__z)
- constexpr complex< float > & operator+= (const complex< _Up > &)
- constexpr complex< float > & operator+= (const float &__t)
- constexpr complex & **operator+=** (float __f)
- template<class _Tp>
  constexpr complex & **operator-=** (const complex< _Tp > &__z)
- constexpr complex< float > & operator-= (const complex< _Up > &)
- constexpr complex< float > & operator-= (const float &__t)
- constexpr complex & **operator-=** (float __f)
- template<class _Tp>
  constexpr complex & **operator/=** (const complex< _Tp > &__z)
- constexpr complex< float > & operator/= (const complex< _Up > &)
- constexpr complex< float > & operator/= (const float &)
- constexpr complex & **operator/=** (float __f)
- constexpr complex & **operator=** (const complex &)=default
- constexpr complex & **operator=** (const complex &)=default
- template<typename _Tp>
  constexpr complex & **operator=** (const complex< _Tp > &__z)
- constexpr complex< float > & operator= (const complex< _Up > &)
- constexpr complex< float > & operator= (const float &)
- constexpr complex & **operator=** (float __f)
- constexpr void **real** (float __val)
- constexpr void **real** (float __val)

5.298.1 Detailed Description

26.2.3 complex specializations complex<float> specialization

5.298.2 Member Typedef Documentation

value_type

```
typedef float std::complex< float >::value_type
```

Value typedef.

5.298.3 Constructor & Destructor Documentation

complex() [1/2]

```
std::complex< float >::complex (
            const float & __r = _Tp(),
            const float & __i = _Tp())  [inline], [constexpr]
```

Default constructor. First parameter is x, second parameter is y. Unspecified parameters default to 0.

complex() [2/2]

```
std::complex< float >::complex (
            const complex< _Up > & __z)  [inline], [constexpr]
```

Converting constructor.

5.298.4 Member Function Documentation

operator*=() [1/2]

```
complex< float > & std::complex< float >::operator*= (
            const complex< _Up > & __z)  [constexpr]
```

Multiply this complex number by another.

operator*=() [2/2]

```
complex< float > & std::complex< float >::operator*= (
            const float & __t)  [constexpr]
```

Multiply this complex number by a scalar.

operator+=() [1/2]

```
complex< float > & std::complex< float >::operator+= (
            const complex< _Up > & __z)  [constexpr]
```

Add another complex number to this one.

operator+=() [2/2]

```
complex< float > & std::complex< float >::operator+= (
            const float & __t)  [inline], [constexpr]
```

Add a scalar to this complex number.

operator-=() [1/2]

```
complex< float > & std::complex< float >::operator-= (
            const complex< _Up > & __z)  [constexpr]
```

Subtract another complex number from this one.

operator-=() [2/2]

```
complex< float > & std::complex< float >::operator-= (
            const float & __t)  [inline], [constexpr]
```

Subtract a scalar from this complex number.

operator/=() [1/2]

```
complex< float > & std::complex< float >::operator/= (
            const complex< _Up > & __z)  [constexpr]
```

Divide this complex number by another.

operator/=() [2/2]

```
complex< float > & std::complex< float >::operator/= (
            const float & __t)  [constexpr]
```

Divide this complex number by a scalar.

operator=() [1/2]

```
complex< float > & std::complex< float >::operator= (
            const complex< _Up > & __z)  [constexpr]
```

Assign another complex number to this one.

operator=() [2/2]

```
complex< float > & std::complex< float >::operator= (
            const float & __t)  [constexpr]
```

Assign a scalar to this complex number.
The documentation for this class was generated from the following file:

- complex

5.299 std::complex< long double > Class Reference

`#include <complex>`

Public Types

- typedef __complex__ long double **_ComplexT**
- typedef long double value_type
- typedef long double **value_type**

Public Member Functions

- constexpr **complex** (_ComplexT __z)
- constexpr **complex** (const complex &)=default
- constexpr **complex** (const complex &)=default
- constexpr complex (const complex< _Up > &__z)
- constexpr **complex** (const complex< double > &__z)
- constexpr **complex** (const complex< float > &__z)
- constexpr complex (const long double &__r=long double(), const long double &__i=long double())
- constexpr **complex** (long double __r=0.0L, long double __i=0.0L)
- **__attribute** ((__abi_tag__("cxx11"))) const expr long double imag() const
- **__attribute** ((__abi_tag__("cxx11"))) const expr long double imag() const
- **__attribute** ((__abi_tag__("cxx11"))) const expr long double real() const
- **__attribute** ((__abi_tag__("cxx11"))) const expr long double real() const
- constexpr complex **__rep** () const
- constexpr _ComplexT **__rep** () const
- constexpr void **imag** (long double __val)
- constexpr void **imag** (long double __val)
- template<typename _Tp>
  constexpr complex & **operator*=** (const complex< _Tp > &__z)
- constexpr complex< long double > & operator*= (const complex< _Up > &)
- constexpr complex< long double > & operator*= (const long double &)
- constexpr complex & **operator*=** (long double __r)
- template<typename _Tp>
  constexpr complex & **operator+=** (const complex< _Tp > &__z)
- constexpr complex< long double > & operator+= (const complex< _Up > &)
- constexpr complex< long double > & operator+= (const long double &__t)
- constexpr complex & **operator+=** (long double __r)

- template<typename _Tp>
  constexpr complex & **operator-=** (const complex< _Tp > &__z)
- constexpr complex< long double > & operator-= (const complex< _Up > &)
- constexpr complex< long double > & operator-= (const long double &__t)
- constexpr complex & **operator-=** (long double __r)
- template<typename _Tp>
  constexpr complex & **operator/=** (const complex< _Tp > &__z)
- constexpr complex< long double > & operator/= (const complex< _Up > &)
- constexpr complex< long double > & operator/= (const long double &)
- constexpr complex & **operator/=** (long double __r)
- constexpr complex & **operator=** (const complex &)=default
- constexpr complex & **operator=** (const complex &)=default
- template<typename _Tp>
  constexpr complex & **operator=** (const complex< _Tp > &__z)
- constexpr complex< long double > & operator= (const complex< _Up > &)
- constexpr complex< long double > & operator= (const long double &)
- constexpr complex & **operator=** (long double __r)
- constexpr void **real** (long double __val)
- constexpr void **real** (long double __val)

5.299.1 Detailed Description

26.2.3 complex specializations complex<long double> specialization

5.299.2 Member Typedef Documentation

value_type

```
typedef long double std::complex< long double >::value_type
```

Value typedef.

5.299.3 Constructor & Destructor Documentation

complex() [1/2]

```
std::complex< long double >::complex (
            const long double & __r = _Tp(),
            const long double & __i = _Tp())  [inline], [constexpr]
```

Default constructor. First parameter is x, second parameter is y. Unspecified parameters default to 0.

complex() [2/2]

```
std::complex< long double >::complex (
            const complex< _Up > & __z)  [inline], [constexpr]
```

Converting constructor.

5.299.4 Member Function Documentation

operator*=() [1/2]

```
complex< long double > & std::complex< long double >::operator*= (
            const complex< _Up > & __z)  [constexpr]
```

Multiply this complex number by another.

operator*=() [2/2]

```
complex< long double > & std::complex< long double >::operator*= (
            const long double & __t)  [constexpr]
```

Multiply this complex number by a scalar.

operator+=() [1/2]

```
complex< long double > & std::complex< long double >::operator+= (
            const complex< _Up > & __z)  [constexpr]
```

Add another complex number to this one.

operator+=() [2/2]

```
complex< long double > & std::complex< long double >::operator+= (
            const long double & __t)  [inline], [constexpr]
```

Add a scalar to this complex number.

operator-=() [1/2]

```
complex< long double > & std::complex< long double >::operator-= (
            const complex< _Up > & __z)  [constexpr]
```

Subtract another complex number from this one.

operator-=() [2/2]

```
complex< long double > & std::complex< long double >::operator-= (
            const long double & __t)  [inline], [constexpr]
```

Subtract a scalar from this complex number.

operator/=() [1/2]

```
complex< long double > & std::complex< long double >::operator/= (
            const complex< _Up > & __z)  [constexpr]
```

Divide this complex number by another.

operator/=() [2/2]

```
complex< long double > & std::complex< long double >::operator/= (
            const long double & __t)  [constexpr]
```

Divide this complex number by a scalar.

operator=() [1/2]

```
complex< long double > & std::complex< long double >::operator= (
            const complex< _Up > & __z)  [constexpr]
```

Assign another complex number to this one.

operator=() [2/2]

```
complex< long double > & std::complex< long double >::operator= (
            const long double & __t)  [constexpr]
```

Assign a scalar to this complex number.

The documentation for this class was generated from the following file:

- complex

5.300 __gnu_pbds::detail::cond_dealtor< Entry, _Alloc > Class Template Reference

`#include <cond_key_dtor_entry_dealtor.hpp>`

Public Types

- typedef HT_Map::entry **entry**
- typedef HT_Map::entry_allocator **entry_allocator**
- typedef alloc_traits::allocator_type **entry_allocator**
- typedef alloc_traits::pointer **entry_pointer**
- typedef HT_Map::key_type **key_type**

Public Member Functions

- **cond_dealtor** (entry_allocator *p_a, entry *p_e)
- **cond_dealtor** (entry_pointer p_e)
- void **set_key_destruct** ()
- void **set_no_action** ()
- void **set_no_action_destructor** ()

Protected Attributes

- bool **m_key_destruct**
- entry_allocator *const **m_p_a**
- entry *const **m_p_e**

5.300.1 Detailed Description

template<typename Entry, typename _Alloc>
class __gnu_pbds::detail::cond_dealtor< Entry, _Alloc >

Conditional deallocate constructor argument.
Conditional dey destructor, cc_hash.
The documentation for this class was generated from the following files:

- cond_dealtor.hpp
- cond_key_dtor_entry_dealtor.hpp

5.301 __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::cond_dtor< Size_Type > Class Template Reference

`#include <ov_tree_map_.hpp>`

Public Member Functions

- **cond_dtor** (value_vector a_vec, iterator &r_last_it, Size_Type total_size)
- void **set_no_action** ()

Protected Attributes

- value_vector **m_a_vec**
- const Size_Type **m_max_size**
- bool **m_no_action**
- iterator & **m_r_last_it**

5.301.1 Detailed Description

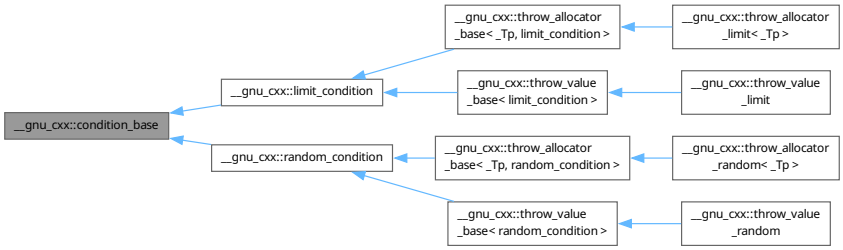
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>
template<typename Size_Type>
class __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::cond_dtor< Size_Type >

Conditional destructor.
The documentation for this class was generated from the following file:

- ov_tree_map_.hpp

5.302 __gnu_cxx::condition_base Struct Reference

`#include <throw_allocator.h>`
Inheritance diagram for __gnu_cxx::condition_base:



Public Member Functions

- **condition_base** (const condition_base &)=default
- condition_base & **operator=** (const condition_base &)=default

5.302.1 Detailed Description

Base struct for condition policy.
Requires a public member function with the signature void throw_conditionally()
The documentation for this struct was generated from the following file:

- throw_allocator.h

5.303 std::condition_variable Class Reference

`#include <condition_variable>`

Public Types

- typedef __gthread_cond_t * **native_handle_type**

Public Member Functions

- **condition_variable** (const condition_variable &)=delete
- native_handle_type **native_handle** ()
- void **notify_all** () noexcept

- void **notify_one** () noexcept
- condition_variable & **operator=** (const condition_variable &)=delete
- void **wait** (unique_lock< mutex > &__lock)
- template<typename _Predicate>
  void **wait** (unique_lock< mutex > &__lock, _Predicate __p)
- template<typename _Rep, typename _Period>
  cv_status **wait_for** (unique_lock< mutex > &__lock, const chrono::duration< _Rep, _Period > &__rtime)
- template<typename _Rep, typename _Period, typename _Predicate>
  bool **wait_for** (unique_lock< mutex > &__lock, const chrono::duration< _Rep, _Period > &__rtime, _Predicate __p)
- template<typename _Clock, typename _Duration>
  cv_status **wait_until** (unique_lock< mutex > &__lock, const chrono::time_point< _Clock, _Duration > &__atime)
- template<typename _Clock, typename _Duration, typename _Predicate>
  bool **wait_until** (unique_lock< mutex > &__lock, const chrono::time_point< _Clock, _Duration > &__atime, _↩ Predicate __p)
- template<typename _Duration>
  cv_status **wait_until** (unique_lock< mutex > &__lock, const chrono::time_point< steady_clock, _Duration > &__atime)
- template<typename _Duration>
  cv_status **wait_until** (unique_lock< mutex > &__lock, const chrono::time_point< system_clock, _Duration > &__atime)

5.303.1 Detailed Description

condition_variable
The documentation for this class was generated from the following file:

- condition_variable

5.304 std::condition_variable_any Class Reference

`#include <condition_variable>`

Public Member Functions

- **condition_variable_any** (const condition_variable_any &)=delete
- void **notify_all** () noexcept
- void **notify_one** () noexcept
- condition_variable_any & **operator=** (const condition_variable_any &)=delete
- template<typename _Lock>
  void **wait** (_Lock &__lock)
- template<typename _Lock, typename _Predicate>
  void **wait** (_Lock &__lock, _Predicate __p)
- template<typename _Lock, typename _Rep, typename _Period>
  cv_status **wait_for** (_Lock &__lock, const chrono::duration< _Rep, _Period > &__rtime)
- template<typename _Lock, typename _Rep, typename _Period, typename _Predicate>
  bool **wait_for** (_Lock &__lock, const chrono::duration< _Rep, _Period > &__rtime, _Predicate __p)
- template<typename _Lock, typename _Clock, typename _Duration>
  cv_status **wait_until** (_Lock &__lock, const chrono::time_point< _Clock, _Duration > &__atime)
- template<typename _Lock, typename _Clock, typename _Duration, typename _Predicate>
  bool **wait_until** (_Lock &__lock, const chrono::time_point< _Clock, _Duration > &__atime, _Predicate __p)

5.304.1 Detailed Description

condition_variable_any
The documentation for this class was generated from the following file:

- condition_variable

5.305 std::conditional< _Cond, _Iftrue, _Iffalse > Struct Template Reference

`#include <type_traits>`

Public Types

- using **type**

5.305.1 Detailed Description

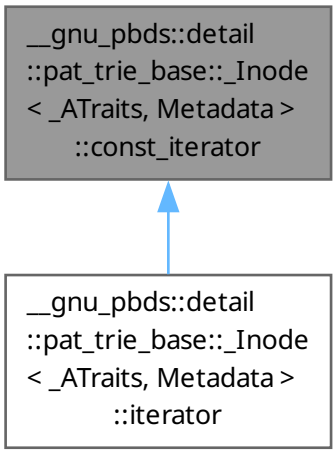
template<bool _Cond, typename _Iftrue, typename _Iffalse>
struct std::conditional< _Cond, _Iftrue, _Iffalse >

Define a member typedef `type` to one of two argument types.
The documentation for this struct was generated from the following file:

- type_traits

5.306 __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >::const_iterator Struct Reference

`#include <pat_trie_base.hpp>`
Inheritance diagram for __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >::const_iterator:



Public Types

- typedef _Alloc::difference_type **difference_type**

- typedef std::forward_iterator_tag **iterator_category**
- typedef node_pointer_pointer **pointer**
- typedef node_pointer_reference **reference**
- typedef node_pointer **value_type**

Public Member Functions

- **const_iterator** (node_pointer_pointer p_p_cur=0, node_pointer_pointer p_p_end=0)
- bool **operator!=** (const const_iterator &other) const
- node_const_pointer **operator*** () const
- const_iterator & **operator++** ()
- const_iterator **operator++** (int)
- const node_pointer_pointer **operator->** () const
- bool **operator==** (const const_iterator &other) const

Public Attributes

- node_pointer_pointer **m_p_p_cur**
- node_pointer_pointer **m_p_p_end**

5.306.1 Detailed Description

template<typename _ATraits, typename Metadata>
struct __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >::const_iterator

Constant child iterator.
The documentation for this struct was generated from the following file:

- pat_trie_base.hpp

5.307 std::chrono::tzdb_list::const_iterator Class Reference

`#include <chrono>`

Public Types

- using **difference_type**
- using **iterator_category**
- using **pointer**
- using **reference**
- using **value_type**

Public Member Functions

- **const_iterator** (const const_iterator &)=default
- **const_iterator** (const_iterator &&)=default
- reference **operator*** () const noexcept
- const_iterator & **operator++** ()
- const_iterator **operator++** (int)
- pointer **operator->** () const noexcept
- const_iterator & **operator=** (const const_iterator &)=default
- const_iterator & **operator=** (const_iterator &&)=default
- bool **operator==** (const const_iterator &) const noexcept=default

Friends

- class **tzdb_list**

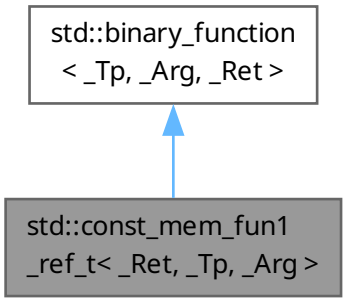
5.307.1 Detailed Description

An iterator into the `tzdb_list`
As a extension, in libstdc++ each `tzdb` is reference-counted and the `const_iterator` type shares ownership of the object it refers to. This ensures that a `tzdb` erased from the list will not be destroyed while there is an iterator that refers to it.
The documentation for this class was generated from the following file:

- chrono

5.308 std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg > Class Template Reference

`#include <stl_function.h>`
Inheritance diagram for std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >:



Public Types

- typedef _Tp first_argument_type
- typedef _Ret result_type
- typedef _Arg second_argument_type

Public Member Functions

- **const_mem_fun1_ref_t** (_Ret(_Tp::*__pf)(_Arg) const)
- _Ret **operator()** (const _Tp &__r, _Arg __x) const

5.308.1 Detailed Description

template<typename _Ret, typename _Tp, typename _Arg>
class std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >

One of the adaptors for member pointers.

5.308.2 Member Typedef Documentation

first_argument_type

`typedef _Tp std::binary_function< _Tp, _Arg, _Ret >::first_argument_type [inherited]`
`first_argument_type` is the type of the first argument

result_type

`typedef _Ret std::binary_function< _Tp, _Arg, _Ret >::result_type [inherited]`
`result_type` is the return type

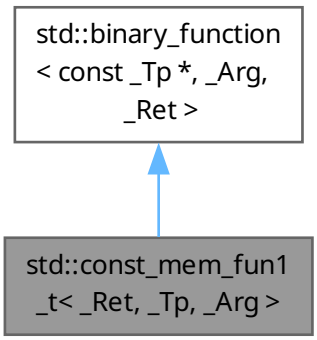
second_argument_type

`typedef _Arg std::binary_function< _Tp, _Arg, _Ret >::second_argument_type [inherited]`
`second_argument_type` is the type of the second argument
The documentation for this class was generated from the following file:

- stl_function.h

5.309 std::const_mem_fun1_t< _Ret, _Tp, _Arg > Class Template Reference

`#include <stl_function.h>`
Inheritance diagram for std::const_mem_fun1_t< _Ret, _Tp, _Arg >:



Public Types

- typedef const _Tp * first_argument_type
- typedef _Ret result_type
- typedef _Arg second_argument_type

Public Member Functions

- **const_mem_fun1_t** (_Ret(_Tp::*__pf)(_Arg) const)
- _Ret **operator()** (const _Tp *__p, _Arg __x) const

5.309.1 Detailed Description

template<typename _Ret, typename _Tp, typename _Arg>
class std::const_mem_fun1_t< _Ret, _Tp, _Arg >

One of the adaptors for member pointers.

5.309.2 Member Typedef Documentation

first_argument_type

```
typedef const _Tp * std::binary_function< const _Tp *, _Arg, _Ret >::first_argument_type  [inherited]
```

`first_argument_type` is the type of the first argument

result_type

```
typedef _Ret std::binary_function< const _Tp *, _Arg, _Ret >::result_type  [inherited]
```

`result_type` is the return type

second_argument_type

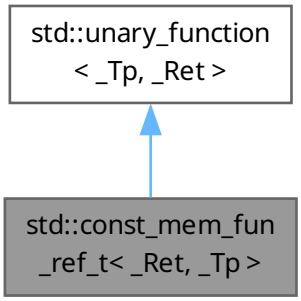
```
typedef _Arg std::binary_function< const _Tp *, _Arg, _Ret >::second_argument_type  [inherited]
```

`second_argument_type` is the type of the second argument
The documentation for this class was generated from the following file:

- stl_function.h

5.310 std::const_mem_fun_ref_t< _Ret, _Tp > Class Template Reference

`#include <stl_function.h>`
Inheritance diagram for std::const_mem_fun_ref_t< _Ret, _Tp >:



Public Types

- typedef _Tp argument_type
- typedef _Ret result_type

Public Member Functions

- **const_mem_fun_ref_t** (_Ret(_Tp::*__pf)() const)
- _Ret **operator()** (const _Tp &__r) const

5.310.1 Detailed Description

template<typename _Ret, typename _Tp>
class std::const_mem_fun_ref_t< _Ret, _Tp >

One of the adaptors for member pointers.

5.310.2 Member Typedef Documentation

argument_type

```
typedef _Tp std::unary_function< _Tp, _Ret >::argument_type  [inherited]
```

`argument_type` is the type of the argument

result_type

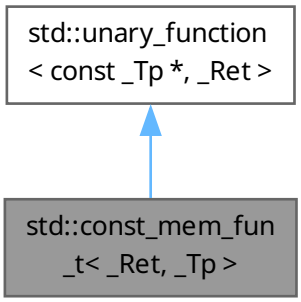
```
typedef _Ret std::unary_function< _Tp, _Ret >::result_type  [inherited]
```

`result_type` is the return type
The documentation for this class was generated from the following file:

- stl_function.h

5.311 std::const_mem_fun_t< _Ret, _Tp > Class Template Reference

`#include <stl_function.h>`
Inheritance diagram for std::const_mem_fun_t< _Ret, _Tp >:



Public Types

- typedef const _Tp * argument_type
- typedef _Ret result_type

Public Member Functions

- **const_mem_fun_t** (_Ret(_Tp::*__pf)() const)
- _Ret **operator()** (const _Tp *__p) const

5.311.1 Detailed Description

template<typename _Ret, typename _Tp>
class std::const_mem_fun_t< _Ret, _Tp >

One of the adaptors for member pointers.

5.311.2 Member Typedef Documentation

argument_type

```
typedef const _Tp * std::unary_function< const _Tp *, _Ret >::argument_type  [inherited]
```

`argument_type` is the type of the argument

result_type

```
typedef _Ret std::unary_function< const _Tp *, _Ret >::result_type  [inherited]
```

`result_type` is the return type
The documentation for this class was generated from the following file:

- stl_function.h

5.312 __gnu_cxx::constant_binary_fun< _Result, _Arg1, _Arg2 > Struct Template Reference

```
#include <functional>
```

Public Types

- typedef _Arg1 **first_argument_type**
- typedef _Result **result_type**
- typedef _Arg2 **second_argument_type**

Public Member Functions

- **constant_binary_fun** (const _Result &__v)
- const result_type & **operator()** (const _Arg1 &, const _Arg2 &) const

Public Attributes

- _Result **_M_val**

5.312.1 Detailed Description

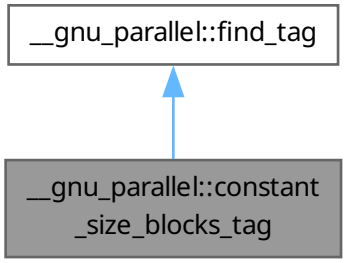
template<class _Result, class _Arg1 = _Result, class _Arg2 = _Arg1>
struct __gnu_cxx::constant_binary_fun< _Result, _Arg1, _Arg2 >

An SGI extension .
The documentation for this struct was generated from the following file:

- ext/functional

5.313 __gnu_parallel::constant_size_blocks_tag Struct Reference

`#include <tags.h>`
Inheritance diagram for __gnu_parallel::constant_size_blocks_tag:



5.313.1 Detailed Description

Selects the constant block size variant for std::find().

See also

_GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS

The documentation for this struct was generated from the following file:

- tags.h

5.314 __gnu_cxx::constant_unary_fun< _Result, _Argument > Struct Template Reference

`#include <functional>`

Public Types

- typedef _Argument **argument_type**
- typedef _Result **result_type**

Public Member Functions

- **constant_unary_fun** (const _Result &__v)
- const result_type & **operator()** (const _Argument &) const

Public Attributes

- result_type **_M_val**

5.314.1 Detailed Description

template<class _Result, class _Argument = _Result>
struct __gnu_cxx::constant_unary_fun< _Result, _Argument >

An SGI extension .
The documentation for this struct was generated from the following file:

- ext/functional

5.315 __gnu_cxx::constant_void_fun< _Result > Struct Template Reference

`#include <functional>`

Public Types

- typedef _Result **result_type**

Public Member Functions

- **constant_void_fun** (const _Result &__v)
- const result_type & **operator()** () const

Public Attributes

- result_type **_M_val**

5.315.1 Detailed Description

template<class _Result>
struct __gnu_cxx::constant_void_fun< _Result >

An SGI extension .
The documentation for this struct was generated from the following file:

- ext/functional

5.316 __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, Tag, Policy_Tl > Struct Template Reference

5.316.1 Detailed Description

template<typename Key, typename Mapped, typename _Alloc, typename Tag, typename Policy_Tl = null_type>
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, Tag, Policy_Tl >

Dispatch mechanism, primary template for associative types.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

5.317 __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binary_heap_tag, null_type > Struct Template Reference

`#include <priority_queue_base_dispatch.hpp>`

Public Types

- typedef binary_heap< _VTp, Cmp_Fn, _Alloc > type

5.317.1 Detailed Description

template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binary_heap_tag, null_type >

Specialization for binary_heap.

5.317.2 Member Typedef Documentation

type

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
typedef binary_heap<_VTp, Cmp_Fn, _Alloc> __gnu_pbds::detail::container_base_dispatch< _VTp,
Cmp_Fn, _Alloc, binary_heap_tag, null_type >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- priority_queue_base_dispatch.hpp

5.318 __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binomial_heap_tag, null_type > Struct Template Reference

```
#include <priority_queue_base_dispatch.hpp>
```

Public Types

- typedef binomial_heap< _VTp, Cmp_Fn, _Alloc > type

5.318.1 Detailed Description

template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binomial_heap_tag, null_type >

Specialization for binomial_heap.

5.318.2 Member Typedef Documentation

type

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
typedef binomial_heap<_VTp, Cmp_Fn, _Alloc> __gnu_pbds::detail::container_base_dispatch< _VTp,
Cmp_Fn, _Alloc, binomial_heap_tag, null_type >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- priority_queue_base_dispatch.hpp

5.319 __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, pairing_heap_tag, null_type > Struct Template Reference

```
#include <priority_queue_base_dispatch.hpp>
```

Public Types

- typedef pairing_heap< _VTp, Cmp_Fn, _Alloc > type

5.319.1 Detailed Description

template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, pairing_heap_tag, null_type >

Specialization for pairing_heap.

5.319.2 Member Typedef Documentation

type

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
typedef pairing_heap<_VTp, Cmp_Fn, _Alloc> __gnu_pbds::detail::container_base_dispatch< _VTp,
Cmp_Fn, _Alloc, pairing_heap_tag, null_type >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- priority_queue_base_dispatch.hpp

5.320 __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type > Struct Template Reference

```
#include <priority_queue_base_dispatch.hpp>
```

Public Types

- typedef rc_binomial_heap< _VTp, Cmp_Fn, _Alloc > type

5.320.1 Detailed Description

template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type >

Specialization for rc_binary_heap.

5.320.2 Member Typedef Documentation

type

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
typedef rc_binomial_heap<_VTp, Cmp_Fn, _Alloc> __gnu_pbds::detail::container_base_dispatch< _VTp,
Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- priority_queue_base_dispatch.hpp

5.321 __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, thin_heap_tag, null_type > Struct Template Reference

```
#include <priority_queue_base_dispatch.hpp>
```

Public Types

- typedef thin_heap< _VTp, Cmp_Fn, _Alloc > type

5.321.1 Detailed Description

template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, thin_heap_tag, null_type >

Specialization for thin_heap.

5.321.2 Member Typedef Documentation

type

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
typedef thin_heap<_VTp, Cmp_Fn, _Alloc> __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_↩
Fn, _Alloc, thin_heap_tag, null_type >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- priority_queue_base_dispatch.hpp

5.322 __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, cc_hash_tag, Policy_Tl > Struct Template Reference

```
#include <container_base_dispatch.hpp>
```

Public Types

- typedef cc_ht_map< Key, Mapped, at0t, at1t, _Alloc, at3t::value, at4t, at2t > type

5.322.1 Detailed Description

template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, cc_hash_tag, Policy_Tl >

Specialization colision-chaining hash map.

5.322.2 Member Typedef Documentation

type

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
typedef cc_ht_map<Key, Mapped, at0t, at1t, _Alloc, at3t::value, at4t, at2t> __gnu_pbds::detail::container_base_d
Key, Mapped, _Alloc, cc_hash_tag, Policy_Tl >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- container_base_dispatch.hpp

5.323 __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, gp_hash_tag, Policy_Tl > Struct Template Reference

```
#include <container_base_dispatch.hpp>
```

Public Types

- typedef gp_ht_map< Key, Mapped, at0t, at1t, _Alloc, at3t::value, at4t, at5t, at2t > type

5.323.1 Detailed Description

template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, gp_hash_tag, Policy_Tl >

Specialization general-probe hash map.

5.323.2 Member Typedef Documentation

type

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
typedef gp_ht_map<Key, Mapped, at0t, at1t, _Alloc, at3t::value, at4t, at5t, at2t> __gnu_pbds::detail::container_b
Key, Mapped, _Alloc, gp_hash_tag, Policy_Tl >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- container_base_dispatch.hpp

5.324 __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, list_update_tag, Policy_Tl > Struct Template Reference

```
#include <container_base_dispatch.hpp>
```

Public Types

- typedef lu_map< Key, Mapped, at0t, _Alloc, at1t > type

5.324.1 Detailed Description

template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, list_update_tag, Policy_Tl >

Specialization for list-update map.

5.324.2 Member Typedef Documentation

type

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
typedef lu_map<Key, Mapped, at0t, _Alloc, at1t> __gnu_pbds::detail::container_base_dispatch< Key,
Mapped, _Alloc, list_update_tag, Policy_Tl >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- container_base_dispatch.hpp

5.325 __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, ov_tree_tag, Policy_Tl > Struct Template Reference

```
#include <container_base_dispatch.hpp>
```

Public Types

- typedef ov_tree_map< Key, Mapped, at0t, at1t, _Alloc > type

5.325.1 Detailed Description

template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, ov_tree_tag, Policy_Tl >

Specialization ordered-vector tree map.

5.325.2 Member Typedef Documentation

type

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
typedef ov_tree_map<Key, Mapped, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch<
Key, Mapped, _Alloc, ov_tree_tag, Policy_Tl >::type
```

Dispatched type.
The documentation for this struct was generated from the following file:

- container_base_dispatch.hpp

5.326 __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, pat_trie_tag, Policy_Tl > Struct Template Reference

```
#include <container_base_dispatch.hpp>
```

Public Types

- typedef pat_trie_map< Key, Mapped, at1t, _Alloc > **type**

5.326.1 Detailed Description

template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, pat_trie_tag, Policy_Tl >

Specialization for PATRICIA trie map.
The documentation for this struct was generated from the following file:

- container_base_dispatch.hpp

5.327 __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, rb_tree_tag, Policy_Tl > Struct Template Reference

```
#include <container_base_dispatch.hpp>
```

Public Types

- typedef rb_tree_map< Key, Mapped, at0t, at1t, _Alloc > type

5.327.1 Detailed Description

template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, rb_tree_tag, Policy_Tl >

Specialization for R-B tree map.

5.327.2 Member Typedef Documentation

type

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
typedef rb_tree_map<Key, Mapped, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch<
Key, Mapped, _Alloc, rb_tree_tag, Policy_Tl >::type
```

Dispatched type.
The documentation for this struct was generated from the following file:

- container_base_dispatch.hpp

5.328 __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, splay_tree_tag, Policy_Tl > Struct Template Reference

`#include <container_base_dispatch.hpp>`

Public Types

- typedef splay_tree_map< Key, Mapped, at0t, at1t, _Alloc > type

5.328.1 Detailed Description

template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, splay_tree_tag, Policy_Tl >

Specialization splay tree map.

5.328.2 Member Typedef Documentation

type

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
typedef splay_tree_map<Key, Mapped, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch<
Key, Mapped, _Alloc, splay_tree_tag, Policy_Tl >::type
```

Dispatched type.
The documentation for this struct was generated from the following file:

- container_base_dispatch.hpp

5.329 __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, cc_hash_tag, Policy_Tl > Struct Template Reference

`#include <container_base_dispatch.hpp>`

Public Types

- typedef cc_ht_set< Key, null_type, at0t, at1t, _Alloc, at3t::value, at4t, at2t > type

5.329.1 Detailed Description

template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, cc_hash_tag, Policy_Tl >

Specialization colision-chaining hash set.

5.329.2 Member Typedef Documentation

type

```
template<typename Key, typename _Alloc, typename Policy_Tl>
typedef cc_ht_set<Key, null_type, at0t, at1t, _Alloc, at3t::value, at4t, at2t> __gnu_pbds::detail::container_base
Key, null_type, _Alloc, cc_hash_tag, Policy_Tl >::type
```

Dispatched type.
The documentation for this struct was generated from the following file:

- container_base_dispatch.hpp

5.330 __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, gp_hash_tag, Policy_Tl > Struct Template Reference

`#include <container_base_dispatch.hpp>`

Public Types

- typedef gp_ht_set< Key, null_type, at0t, at1t, _Alloc, at3t::value, at4t, at5t, at2t > type

5.330.1 Detailed Description

template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, gp_hash_tag, Policy_Tl >

Specialization general-probe hash set.

5.330.2 Member Typedef Documentation

type

```
template<typename Key, typename _Alloc, typename Policy_Tl>
typedef gp_ht_set<Key, null_type, at0t, at1t, _Alloc, at3t::value, at4t, at5t, at2t> __gnu_pbds::detail::containe
Key, null_type, _Alloc, gp_hash_tag, Policy_Tl >::type
```

Dispatched type.
The documentation for this struct was generated from the following file:

- container_base_dispatch.hpp

5.331 __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, list_update_tag, Policy_Tl > Struct Template Reference

`#include <container_base_dispatch.hpp>`

Public Types

- typedef lu_set< Key, null_type, at0t, _Alloc, at1t > type

5.331.1 Detailed Description

template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, list_update_tag, Policy_Tl >

Specialization for list-update set.

5.331.2 Member Typedef Documentation

type

```
template<typename Key, typename _Alloc, typename Policy_Tl>
typedef lu_set<Key, null_type, at0t, _Alloc, at1t> __gnu_pbds::detail::container_base_dispatch<
Key, null_type, _Alloc, list_update_tag, Policy_Tl >::type
```

Dispatched type.
The documentation for this struct was generated from the following file:

- container_base_dispatch.hpp

5.332 __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, ov_tree_tag, Policy_Tl > Struct Template Reference

`#include <container_base_dispatch.hpp>`

Public Types

- typedef ov_tree_set< Key, null_type, at0t, at1t, _Alloc > type

5.332.1 Detailed Description

template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, ov_tree_tag, Policy_Tl >

Specialization ordered-vector tree set.

5.332.2 Member Typedef Documentation

type

```
template<typename Key, typename _Alloc, typename Policy_Tl>
typedef ov_tree_set<Key, null_type, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch<
Key, null_type, _Alloc, ov_tree_tag, Policy_Tl >::type
```

Dispatched type.
The documentation for this struct was generated from the following file:

- container_base_dispatch.hpp

5.333 __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, pat_trie_tag, Policy_Tl > Struct Template Reference

`#include <container_base_dispatch.hpp>`

Public Types

- typedef pat_trie_set< Key, null_type, at1t, _Alloc > type

5.333.1 Detailed Description

template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, pat_trie_tag, Policy_Tl >

Specialization for PATRICIA trie set.

5.333.2 Member Typedef Documentation

type

```
template<typename Key, typename _Alloc, typename Policy_Tl>
typedef pat_trie_set<Key, null_type, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch<
Key, null_type, _Alloc, pat_trie_tag, Policy_Tl >::type
```

Dispatched type.
The documentation for this struct was generated from the following file:

- container_base_dispatch.hpp

5.334 __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, rb_tree_tag, Policy_Tl > Struct Template Reference

`#include <container_base_dispatch.hpp>`

Public Types

- typedef rb_tree_set< Key, null_type, at0t, at1t, _Alloc > **type**

5.334.1 Detailed Description

template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, rb_tree_tag, Policy_Tl >

Specialization for R-B tree set.
The documentation for this struct was generated from the following file:

- container_base_dispatch.hpp

5.335 __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, splay_tree_tag, Policy_Tl > Struct Template Reference

`#include <container_base_dispatch.hpp>`

Public Types

- typedef splay_tree_set< Key, null_type, at0t, at1t, _Alloc > type

5.335.1 Detailed Description

template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, splay_tree_tag, Policy_Tl >

Specialization splay tree set.

5.335.2 Member Typedef Documentation

type

```
template<typename Key, typename _Alloc, typename Policy_Tl>
typedef splay_tree_set<Key, null_type, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch<
Key, null_type, _Alloc, splay_tree_tag, Policy_Tl >::type
```

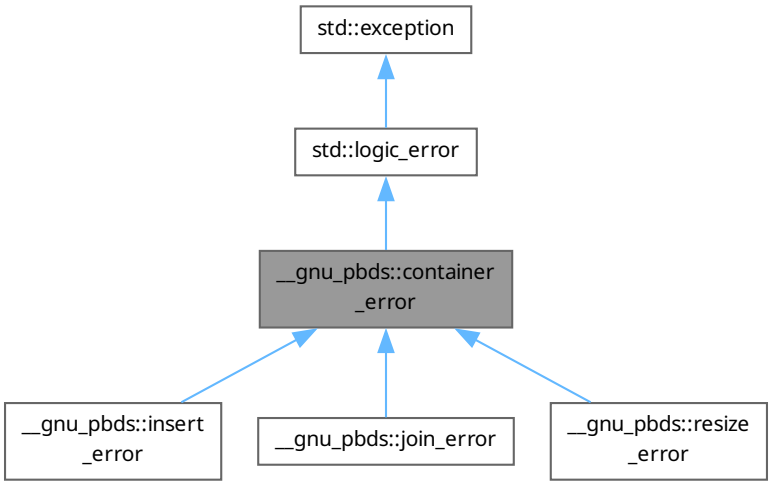
Dispatched type.
The documentation for this struct was generated from the following file:

- container_base_dispatch.hpp

5.336 __gnu_pbds::container_error Struct Reference

`#include <exception.hpp>`

Inheritance diagram for __gnu_pbds::container_error:



Public Member Functions

- virtual const char ∗ what () const noexcept

5.336.1 Detailed Description

Base class for exceptions.

5.336.2 Member Function Documentation

what()

```
virtual const char * std::logic_error::what () const  [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).
Reimplemented from std::exception.
Reimplemented in std::future_error.
The documentation for this struct was generated from the following file:

- exception.hpp

5.337 __gnu_pbds::container_tag Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for __gnu_pbds::container_tag:

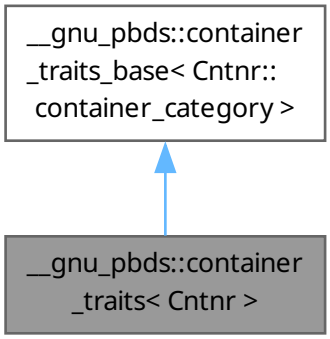
5.337.1 Detailed Description

Base data structure tag.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

5.338 __gnu_pbds::container_traits< Cntnr > Struct Template Reference

`#include <tag_and_trait.hpp>`
Inheritance diagram for __gnu_pbds::container_traits< Cntnr >:



Public Types

- enum { order_preserving , erase_can_throw , split_join_can_throw , reverse_iteration }
- typedef container_traits_base< container_category > **base_type**
- typedef Cntnr::container_category **container_category**
- typedef Cntnr **container_type**
- typedef base_type::invalidation_guarantee **invalidation_guarantee**

5.338.1 Detailed Description

template<typename Cntnr>
struct __gnu_pbds::container_traits< Cntnr >

Container traits.

5.338.2 Member Enumeration Documentation

anonymous enum

```
template<typename Cntnr>
anonymous enum
```

Enumerator

order_preserving	True only if Cntnr objects guarantee storing keys by order.
erase_can_throw	True only if erasing a key can throw.
split_join_can_throw	True only if split or join operations can throw.
reverse_iteration	True only reverse iterators are supported.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

5.339 __gnu_pbds::container_traits_base< _Tag > Struct Template Reference

5.339.1 Detailed Description

template<typename _Tag>
struct __gnu_pbds::container_traits_base< _Tag >

Primary template, container traits base.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

5.340 __gnu_pbds::container_traits_base< binary_heap_tag > Struct Reference

```
#include <tag_and_trait.hpp>
```

Public Types

- enum { **order_preserving** , **erase_can_throw** , **split_join_can_throw** , **reverse_iteration** }
- typedef binary_heap_tag **container_category**
- typedef basic_invalidation_guarantee **invalidation_guarantee**

5.340.1 Detailed Description

Specialization, binary heap.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

5.341 __gnu_pbds::container_traits_base< binomial_heap_tag > Struct Reference

`#include <tag_and_trait.hpp>`

Public Types

- enum { **order_preserving** , **erase_can_throw** , **split_join_can_throw** , **reverse_iteration** }
- typedef binomial_heap_tag **container_category**
- typedef point_invalidation_guarantee **invalidation_guarantee**

5.341.1 Detailed Description

Specialization, binomial heap.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

5.342 __gnu_pbds::container_traits_base< cc_hash_tag > Struct Reference

`#include <tag_and_trait.hpp>`

Public Types

- enum { **order_preserving** , **erase_can_throw** , **split_join_can_throw** , **reverse_iteration** }
- typedef cc_hash_tag **container_category**
- typedef point_invalidation_guarantee **invalidation_guarantee**

5.342.1 Detailed Description

Specialization, cc hash.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

5.343 __gnu_pbds::container_traits_base< gp_hash_tag > Struct Reference

`#include <tag_and_trait.hpp>`

Public Types

- enum { **order_preserving** , **erase_can_throw** , **split_join_can_throw** , **reverse_iteration** }
- typedef gp_hash_tag **container_category**
- typedef basic_invalidation_guarantee **invalidation_guarantee**

5.343.1 Detailed Description

Specialization, gp hash.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

5.344 __gnu_pbds::container_traits_base< list_update_tag > Struct Reference

`#include <tag_and_trait.hpp>`

Public Types

- enum { **order_preserving** , **erase_can_throw** , **split_join_can_throw** , **reverse_iteration** }
- typedef list_update_tag **container_category**
- typedef point_invalidation_guarantee **invalidation_guarantee**

5.344.1 Detailed Description

Specialization, list update.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

5.345 __gnu_pbds::container_traits_base< ov_tree_tag > Struct Reference

`#include <tag_and_trait.hpp>`

Public Types

- enum { **order_preserving** , **erase_can_throw** , **split_join_can_throw** , **reverse_iteration** }
- typedef ov_tree_tag **container_category**
- typedef basic_invalidation_guarantee **invalidation_guarantee**

5.345.1 Detailed Description

Specialization, ov tree.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

5.346 __gnu_pbds::container_traits_base< pairing_heap_tag > Struct Reference

`#include <tag_and_trait.hpp>`

Public Types

- enum { **order_preserving** , **erase_can_throw** , **split_join_can_throw** , **reverse_iteration** }
- typedef pairing_heap_tag **container_category**
- typedef point_invalidation_guarantee **invalidation_guarantee**

5.346.1 Detailed Description

Specialization, pairing heap.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

5.347 __gnu_pbds::container_traits_base< pat_trie_tag > Struct Reference

`#include <tag_and_trait.hpp>`

Public Types

- enum { **order_preserving** , **erase_can_throw** , **split_join_can_throw** , **reverse_iteration** }
- typedef pat_trie_tag **container_category**
- typedef range_invalidation_guarantee **invalidation_guarantee**

5.347.1 Detailed Description

Specialization, pat trie.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

5.348 __gnu_pbds::container_traits_base< rb_tree_tag > Struct Reference

`#include <tag_and_trait.hpp>`

Public Types

- enum { **order_preserving** , **erase_can_throw** , **split_join_can_throw** , **reverse_iteration** }
- typedef rb_tree_tag **container_category**
- typedef range_invalidation_guarantee **invalidation_guarantee**

5.348.1 Detailed Description

Specialization, rb tree.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

5.349 __gnu_pbds::container_traits_base< rc_binomial_heap_tag > Struct Reference

`#include <tag_and_trait.hpp>`

Public Types

- enum { **order_preserving** , **erase_can_throw** , **split_join_can_throw** , **reverse_iteration** }
- typedef rc_binomial_heap_tag **container_category**
- typedef point_invalidation_guarantee **invalidation_guarantee**

5.349.1 Detailed Description

Specialization, rc binomial heap.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

5.350 __gnu_pbds::container_traits_base< splay_tree_tag > Struct Reference

`#include <tag_and_trait.hpp>`

Public Types

- enum { **order_preserving** , **erase_can_throw** , **split_join_can_throw** , **reverse_iteration** }
- typedef splay_tree_tag **container_category**
- typedef range_invalidation_guarantee **invalidation_guarantee**

5.350.1 Detailed Description

Specialization, splay tree.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

5.351 __gnu_pbds::container_traits_base< thin_heap_tag > Struct Reference

`#include <tag_and_trait.hpp>`

Public Types

- enum { **order_preserving** , **erase_can_throw** , **split_join_can_throw** , **reverse_iteration** }
- typedef thin_heap_tag **container_category**
- typedef point_invalidation_guarantee **invalidation_guarantee**

5.351.1 Detailed Description

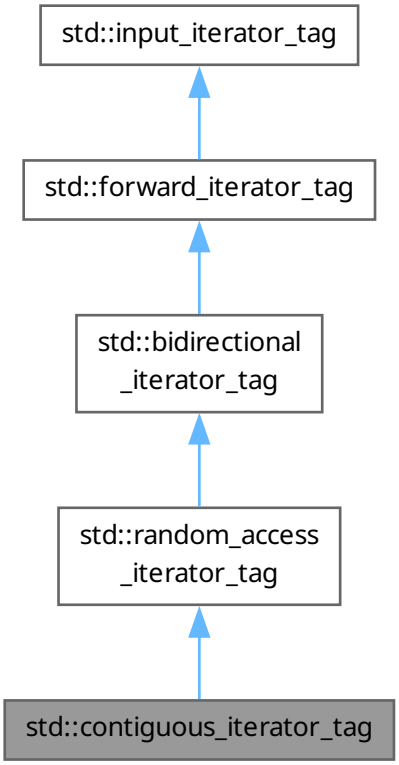
Specialization, thin heap.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

5.352 std::contiguous_iterator_tag Struct Reference

`#include <stl_iterator_base_types.h>`

Inheritance diagram for std::contiguous_iterator_tag:



5.352.1 Detailed Description

Contiguous iterators point to objects stored contiguously in memory.
The documentation for this struct was generated from the following file:

- stl_iterator_base_types.h

5.353 std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> Class Template Reference

`#include <functional>`

Public Types

- using **result_type**

Public Member Functions

- copyable_function () noexcept

- template<typename _Fn, typename _Vt = decay_t<_Fn>>
  requires (!is_same_v<_Vt, copyable_function>) && (!__is_in_place_type_v<_Vt>) && __is_callable_from<_Vt>
  copyable_function (_Fn &&__f) noexcept(_S_nothrow_init< _Vt, _Fn >())
- copyable_function (copyable_function &&__x) noexcept
- copyable_function (copyable_function const &__x)
- template<typename _Tp, typename... _Args>
  requires is_constructible_v<_Tp, _Args...> && __is_callable_from<_Tp>
  copyable_function (in_place_type_t< _Tp >, _Args &&... __args) noexcept(_S_nothrow_init< _Tp, _Args... >())
- template<typename _Tp, typename _Up, typename... _Args>
  requires is_constructible_v<_Tp, initializer_list<_Up>&, _Args...> && __is_callable_from<_Tp>
  copyable_function (in_place_type_t< _Tp >, initializer_list< _Up > __il, _Args &&... __args) noexcept(_S_↩
  nothrow_init< _Tp, initializer_list< _Up > &, _Args... >())
- copyable_function (nullptr_t) noexcept
- operator bool () const noexcept
- _Res operator() (_ArgTypes... __args) _GLIBCXX_MOF_CV noexcept(_Noex)
- template<typename _Fn>
  requires is_constructible_v<copyable_function, _Fn>
  copyable_function & operator= (_Fn &&__f) noexcept(is_nothrow_constructible_v< copyable_function, _Fn >)
- copyable_function & operator= (const copyable_function &__x)
- copyable_function & operator= (copyable_function &&__x) noexcept
- copyable_function & operator= (nullptr_t) noexcept
- void swap (copyable_function &__x) noexcept

Friends

- template<typename _Func>
  auto & **__polyfunc::__base_of** (_Func &) noexcept
- template<typename _Func>
  auto & **__polyfunc::__invoker_of** (_Func &) noexcept
- template<typename _Dst, typename _Src>
  consteval bool **__polyfunc::__is_invoker_convertible** () noexcept
- bool operator== (const copyable_function &__x, nullptr_t) noexcept
- void swap (copyable_function &__x, copyable_function &__y) noexcept

5.353.1 Detailed Description

template<typename _Res, typename... _ArgTypes, bool _Noex>
class std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>

Polymorphic copyable function wrapper.

Since

C++26

The `std::copyable_function` class template is a call wrapper similar to `std::function`, but it does not provide information about it's target, and preserves constness.
It also supports const-qualification, ref-qualification, and no-throw guarantees. The qualifications and exception-specification of the `copyable_function::operator()` member function are respected when invoking the target function.

5.353.2 Constructor & Destructor Documentation

copyable_function() [1/7]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::copyable_function
() [inline], [noexcept]
```

Creates an empty object.

References copyable_function().

Referenced by copyable_function(), copyable_function(), copyable_function(), copyable_function(), copyable_function(), copyable_function(), copyable_function(), operator=(), operator=(), operator=(), operator=(), operator==, swap(), and swap.

copyable_function() [2/7]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::copyable_function (
            nullptr_t ) [inline], [noexcept]
```

Creates an empty object.

References copyable_function().

copyable_function() [3/7]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::copyable_function (
            copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> && __x)
[inline], [noexcept]
```

Moves the target object, leaving the source empty.

References copyable_function().

copyable_function() [4/7]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::copyable_function (
            copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> const & __x)
[inline]
```

Copies the target object.

References copyable_function().

copyable_function() [5/7]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<typename _Fn, typename _Vt = decay_t<_Fn>>
requires (!is_same_v<_Vt, copyable_function>) && (!__is_in_place_type_v<_Vt>) && __is_callable↩
_from<_Vt>
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::copyable_function (
            _Fn && __f) [inline], [noexcept]
```

Stores a target object initialized from the argument.

References copyable_function(), and std::forward().

copyable_function() [6/7]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<typename _Tp, typename... _Args>
requires is_constructible_v<_Tp, _Args...> && __is_callable_from<_Tp>
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::copyable_function (
```

```
            in_place_type_t< _Tp > ,
            _Args &&...  __args)  [inline], [explicit], [noexcept]
```

Stores a target object initialized from the arguments.
References copyable_function(), and std::forward().

copyable_function() [7/7]

```
template<typename _Res, typename...  _ArgTypes, bool _Noex>
template<typename _Tp, typename _Up, typename...  _Args>
requires is_constructible_v<_Tp, initializer_list<_Up>&, _Args...> && __is_callable_from<_Tp>
std::copyable_function< _Res(_ArgTypes...)  _GLIBCXX_MOF_CV noexcept(_Noex)>::copyable_function (
            in_place_type_t< _Tp > ,
            initializer_list< _Up > __il,
            _Args &&...  __args)  [inline], [explicit], [noexcept]
```

Stores a target object initialized from the arguments.
References copyable_function(), and std::forward().

5.353.3 Member Function Documentation

operator bool()

```
template<typename _Res, typename...  _ArgTypes, bool _Noex>
std::copyable_function< _Res(_ArgTypes...)  _GLIBCXX_MOF_CV noexcept(_Noex)>::operator bool ()
const  [inline], [explicit], [noexcept]
```

True if a target object is present, false otherwise.

operator()()

```
template<typename _Res, typename...  _ArgTypes, bool _Noex>
_Res std::copyable_function< _Res(_ArgTypes...)  _GLIBCXX_MOF_CV noexcept(_Noex)>::operator() (
            _ArgTypes...  __args)  [inline], [noexcept]
```

Invoke the target object.
The target object will be invoked using the supplied arguments, and as an lvalue or rvalue, and as const or non-const, as dictated by the template arguments of the `copyable_function` specialization.

Precondition

Must not be empty.

References std::forward(), and operator()().
Referenced by operator()().

operator=() [1/4]

```
template<typename _Res, typename...  _ArgTypes, bool _Noex>
template<typename _Fn>
requires is_constructible_v<copyable_function, _Fn>
copyable_function & std::copyable_function< _Res(_ArgTypes...)  _GLIBCXX_MOF_CV noexcept(_↩
Noex)>::operator= (
            _Fn && __f)  [inline], [noexcept]
```

Stores a new target object, initialized from the argument.
References copyable_function(), std::forward(), and operator=().

operator=() [2/4]

```
template<typename _Res, typename...  _ArgTypes, bool _Noex>
copyable_function & std::copyable_function< _Res(_ArgTypes...)  _GLIBCXX_MOF_CV noexcept(_↩
```

```
Noex)>::operator= (
            const copyable_function< _Res(_ArgTypes...)  _GLIBCXX_MOF_CV noexcept(_Noex)> & __x)
[inline]
```

Stores a copy of the source target object.
References copyable_function(), and operator=().

operator=() [3/4]

```
template<typename _Res, typename...  _ArgTypes, bool _Noex>
copyable_function & std::copyable_function< _Res(_ArgTypes...)  _GLIBCXX_MOF_CV noexcept(_↩
Noex)>::operator= (
            copyable_function< _Res(_ArgTypes...)  _GLIBCXX_MOF_CV noexcept(_Noex)> && __x)
[inline], [noexcept]
```

Stores a new target object, leaving `x` empty.
References copyable_function(), std::addressof(), and operator=().
Referenced by operator=(), operator=(), operator=(), and operator=().

operator=() [4/4]

```
template<typename _Res, typename...  _ArgTypes, bool _Noex>
copyable_function & std::copyable_function< _Res(_ArgTypes...)  _GLIBCXX_MOF_CV noexcept(_↩
Noex)>::operator= (
            nullptr_t )  [inline], [noexcept]
```

Destroys the target object (if any).
References copyable_function(), and operator=().

swap()

```
template<typename _Res, typename...  _ArgTypes, bool _Noex>
void std::copyable_function< _Res(_ArgTypes...)  _GLIBCXX_MOF_CV noexcept(_Noex)>::swap (
            copyable_function< _Res(_ArgTypes...)  _GLIBCXX_MOF_CV noexcept(_Noex)> & __x)  [inline],
[noexcept]
```

Exchange the target objects (if any).
References copyable_function(), and swap().
Referenced by swap(), and swap.

5.353.4 Friends And Related Symbol Documentation

operator==

```
template<typename _Res, typename...  _ArgTypes, bool _Noex>
bool operator== (
            const copyable_function< _Res(_ArgTypes...)  _GLIBCXX_MOF_CV noexcept(_Noex)> & __x,
            nullptr_t )  [friend]
```

Check for emptiness by comparing with `nullptr`.
References copyable_function(), and operator==.
Referenced by operator==.

swap

```
template<typename _Res, typename...  _ArgTypes, bool _Noex>
void swap (
            copyable_function< _Res(_ArgTypes...)  _GLIBCXX_MOF_CV noexcept(_Noex)> & __x,
            copyable_function< _Res(_ArgTypes...)  _GLIBCXX_MOF_CV noexcept(_Noex)> & __y)  [friend]
```

Exchange the target objects (if any).

References copyable_function(), and swap().
The documentation for this class was generated from the following file:

- cpyfunc_impl.h

5.354 std::counted_iterator< _It > Class Template Reference

`#include <stl_iterator.h>`

Public Types

- using **difference_type**
- using **iterator_type**

Public Member Functions

- constexpr **counted_iterator** (_It __i, iter_difference_t< _It > __n)
- template<typename _It2>
  requires convertible_to<const _It2&, _It>
  constexpr **counted_iterator** (const counted_iterator< _It2 > &__x)
- constexpr _It **base** () &&noexcept(is_nothrow_move_constructible_v< _It >)
- constexpr const _It & **base** () const &noexcept
- constexpr iter_difference_t< _It > **count** () const noexcept
- constexpr decltype(auto) **operator*** () const noexcept(noexcept(*_M_current))
- constexpr decltype(auto) **operator*** () noexcept(noexcept(*_M_current))
- constexpr counted_iterator **operator+** (iter_difference_t< _It > __n) const
- constexpr counted_iterator & **operator++** ()
- constexpr decltype(auto) **operator++** (int)
- constexpr counted_iterator **operator++** (int)
- constexpr counted_iterator & **operator+=** (iter_difference_t< _It > __n)
- constexpr counted_iterator **operator-** (iter_difference_t< _It > __n) const
- constexpr counted_iterator & **operator--** ()
- constexpr counted_iterator **operator--** (int)
- constexpr counted_iterator & **operator-=** (iter_difference_t< _It > __n)
- constexpr auto **operator->** () const noexcept
- template<typename _It2>
  requires assignable_from<_It&, const _It2&>
  constexpr counted_iterator & **operator=** (const counted_iterator< _It2 > &__x)
- constexpr decltype(auto) **operator[ ]** (iter_difference_t< _It > __n) const noexcept(noexcept(_M_current[__n]))

Friends

- constexpr iter_rvalue_reference_t< _It > **iter_move** (const counted_iterator &__i) noexcept(noexcept(ranges↩
  ::iter_move(__i._M_current)))
- template<indirectly_swappable< _It > _It2>
  constexpr void **iter_swap** (const counted_iterator &__x, const counted_iterator< _It2 > &__y) noexcept(noexcept(ranges↩
  ::iter_swap(__x._M_current, __y._M_current)))
- constexpr counted_iterator **operator+** (iter_difference_t< _It > __n, const counted_iterator &__x)
- template<common_with< _It > _It2>
  constexpr iter_difference_t< _It2 > **operator-** (const counted_iterator &__x, const counted_iterator< _It2 > &↩
  __y) noexcept
- constexpr iter_difference_t< _It > **operator-** (const counted_iterator &__x, default_sentinel_t) noexcept
- constexpr iter_difference_t< _It > **operator-** (default_sentinel_t, const counted_iterator &__y) noexcept

- template<common_with< _It > _It2>
  constexpr strong_ordering **operator<=>** (const counted_iterator &__x, const counted_iterator< _It2 > &__y) noexcept
- template<common_with< _It > _It2>
  constexpr bool **operator==** (const counted_iterator &__x, const counted_iterator< _It2 > &__y) noexcept
- constexpr bool **operator==** (const counted_iterator &__x, default_sentinel_t) noexcept

5.354.1 Detailed Description

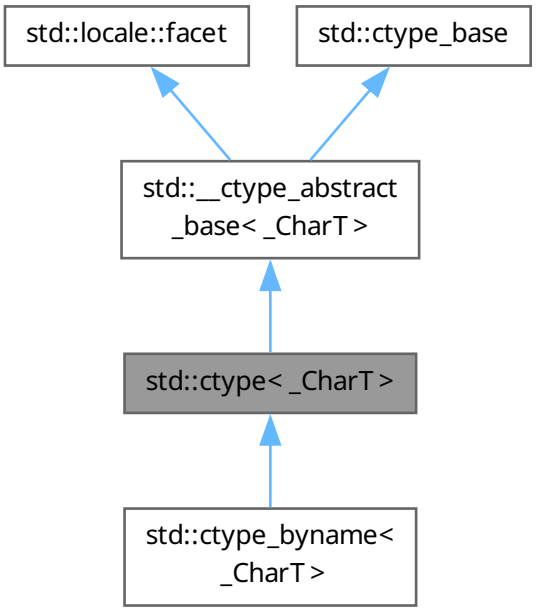
template<input_or_output_iterator _It>
class std::counted_iterator< _It >

An iterator adaptor that keeps track of the distance to the end.
The documentation for this class was generated from the following file:

- bits/stl_iterator.h

5.355 std::ctype< _CharT > Class Template Reference

`#include <locale_facets.h>`
Inheritance diagram for std::ctype< _CharT >:



Public Types

- typedef const int * **__to_type**
- typedef _CharT **char_type**
- typedef __ctype_abstract_base< _CharT >::mask **mask**

Public Member Functions

- **ctype** (size_t __refs=0)
- const char_type * is (const char_type *__lo, const char_type *__hi, mask *__vec) const
- bool is (mask __m, char_type __c) const
- char narrow (char_type __c, char __dfault) const
- const char_type * narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__to) const
- const char_type * scan_is (mask __m, const char_type *__lo, const char_type *__hi) const
- const char_type * scan_not (mask __m, const char_type *__lo, const char_type *__hi) const
- const char_type * tolower (char_type *__lo, const char_type *__hi) const
- char_type tolower (char_type __c) const
- const char_type * toupper (char_type *__lo, const char_type *__hi) const
- char_type toupper (char_type __c) const
- char_type widen (char __c) const
- const char * widen (const char *__lo, const char *__hi, char_type *__to) const

Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **blank**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static locale::id id
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

Protected Member Functions

- virtual const char_type * do_is (const char_type *__lo, const char_type *__hi, mask *__vec) const
- virtual bool do_is (mask __m, char_type __c) const
- virtual char do_narrow (char_type, char __dfault) const
- virtual const char_type * do_narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__to) const
- virtual const char_type * do_scan_is (mask __m, const char_type *__lo, const char_type *__hi) const
- virtual const char_type * do_scan_not (mask __m, const char_type *__lo, const char_type *__hi) const
- virtual const char_type * do_tolower (char_type *__lo, const char_type *__hi) const
- virtual char_type do_tolower (char_type __c) const
- virtual const char_type * do_toupper (char_type *__lo, const char_type *__hi) const
- virtual char_type do_toupper (char_type __c) const
- virtual char_type do_widen (char __c) const
- virtual const char * do_widen (const char *__lo, const char *__hi, char_type *__dest) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

5.355.1 Detailed Description

template<typename _CharT>
class std::ctype< _CharT >

Primary class template ctype facet.
This template class defines classification and conversion functions for character sets. It wraps cctype functionality. Ctype gets used by streams for many I/O operations.
This template provides the protected virtual functions the developer will have to replace in a derived class or specialization to make a working facet. The public functions that access them are defined in __ctype_abstract_base, to allow for implementation flexibility. See ctype<wchar_t> for an example. The functions are documented in __ctype_abstract_base.
Note: implementations are provided for all the protected virtual functions, but will likely not be useful.

5.355.2 Member Function Documentation

do_is() [1/2]

```
template<typename _CharT>
virtual const char_type * std::ctype< _CharT >::do_is (
            const char_type * __lo,
            const char_type * __hi,
            mask * __vec) const  [protected], [virtual]
```

Return a mask array.
This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.
do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__vec	Pointer to an array of mask storage.

Returns

__hi.

Implements std::__ctype_abstract_base< _CharT >.

do_is() [2/2]

```
template<typename _CharT>
virtual bool std::ctype< _CharT >::do_is (
```

```
            mask __m,
            char_type __c) const  [protected], [virtual]
```

Test char_type classification.
This function finds a mask M for *c* and compares it to mask *m*.
do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

_↩ _c	The char_type to find the mask of.
_↩ _m	The mask to compare against.

Returns

(M & __m) != 0.

Implements std::__ctype_abstract_base< _CharT >.

do_narrow() [1/2]

```
template<typename _CharT>
virtual char std::ctype< _CharT >::do_narrow (
            char_type __c,
            char __dfault) const  [protected], [virtual]
```

Narrow char_type to char.
This virtual function converts the argument to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead.
do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char_type to convert.
__dfault	Char to return if conversion fails.

Returns

The converted char.

Implements std::__ctype_abstract_base< _CharT >.
Referenced by std::ctype< char >::narrow(), and std::ctype< char >::narrow().

do_narrow() [2/2]

```
template<typename _CharT>
virtual const char_type * std::ctype< _CharT >::do_narrow (
            const char_type * __lo,
            const char_type * __hi,
            char __dfault,
            char * __to) const  [protected], [virtual]
```

Narrow char_type array to char.

This virtual function converts each char_type in the range [__lo,__hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, *__dfault* is used instead.
do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

Implements std::__ctype_abstract_base< _CharT >.

do_scan_is()

```
template<typename _CharT>
virtual const char_type * std::ctype< _CharT >::do_scan_is (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [protected], [virtual]
```

Find char_type matching mask.
This function searches for and returns the first char_type c in [__lo,__hi) for which is(__m,c) is true.
do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

__m	The mask to compare against.
__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

Pointer to a matching char_type if found, else *__hi*.

Implements std::__ctype_abstract_base< _CharT >.

do_scan_not()

```
template<typename _CharT>
virtual const char_type * std::ctype< _CharT >::do_scan_not (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [protected], [virtual]
```

Find char_type not matching mask.
This function searches for and returns a pointer to the first char_type c of [lo,hi) for which is(m,c) is false.
do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to a non-matching char_type if found, else *__hi*.

Implements std::__ctype_abstract_base< _CharT >.

do_tolower() [1/2]

```
template<typename _CharT>
virtual const char_type * std::ctype< _CharT >::do_tolower (
            char_type * __lo,
            const char_type * __hi) const  [protected], [virtual]
```

Convert array to lowercase.
This virtual function converts each char_type in the range [__lo,__hi) to lowercase if possible. Other elements remain untouched.
do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

Implements std::__ctype_abstract_base< _CharT >.

do_tolower() [2/2]

```
template<typename _CharT>
virtual char_type std::ctype< _CharT >::do_tolower (
            char_type __c) const  [protected], [virtual]
```

Convert to lowercase.
This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.
do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

_↩ _c	The char_type to convert.

Returns

The lowercase char_type if convertible, else __c.

Implements std::__ctype_abstract_base< _CharT >.
Referenced by std::ctype< char >::tolower(), and std::ctype< char >::tolower().

do_toupper() [1/2]

```
template<typename _CharT>
virtual const char_type * std::ctype< _CharT >::do_toupper (
            char_type * __lo,
            const char_type * __hi) const  [protected], [virtual]
```

Convert array to uppercase.
This virtual function converts each char_type in the range [__lo,__hi) to uppercase if possible. Other elements remain untouched.
do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

Implements std::__ctype_abstract_base< _CharT >.

do_toupper() [2/2]

```
template<typename _CharT>
virtual char_type std::ctype< _CharT >::do_toupper (
            char_type __c) const  [protected], [virtual]
```

Convert to uppercase.
This virtual function converts the char_type argument to uppercase if possible. If not possible (for example, '2'), returns the argument.
do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

_↩ _c	The char_type to convert.

Returns

The uppercase char_type if convertible, else *__c*.

Implements std::__ctype_abstract_base< _CharT >.
Referenced by std::ctype< char >::toupper(), and std::ctype< char >::toupper().

do_widen() [1/2]

```
template<typename _CharT>
virtual char_type std::ctype< _CharT >::do_widen (
            char __c) const  [protected], [virtual]
```

Widen char.
This virtual function converts the char to char_type using the simplest reasonable transformation.
do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char to convert.

Returns

The converted char_type

Implements std::__ctype_abstract_base< _CharT >.
Referenced by std::ctype< char >::widen(), and std::ctype< char >::widen().

do_widen() [2/2]

```
template<typename _CharT>
virtual const char * std::ctype< _CharT >::do_widen (
            const char * __lo,
            const char * __hi,
            char_type * __to) const  [protected], [virtual]
```

Widen char array.
This function converts each char in the input to char_type using the simplest reasonable transformation.
do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start range.
__hi	Pointer to end of range.
__to	Pointer to the destination array.

Returns

__hi.

Implements std::__ctype_abstract_base< _CharT >.

is() [1/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::is (
            const char_type * __lo,
            const char_type * __hi,
            mask * __vec) const  [inline], [inherited]
```

Return a mask array.
This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char_type>::do_is().

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__vec	Pointer to an array of mask storage.

Returns

__hi.

is() [2/2]

```
template<typename _CharT>
bool std::__ctype_abstract_base< _CharT >::is (
            mask __m,
            char_type __c) const  [inline], [inherited]
```

Test char_type classification.
This function finds a mask M for *__c* and compares it to mask *__m*. It does so by returning the value of ctype<char_↩ type>::do_is().

Parameters

_↩ _c	The char_type to compare the mask of.
_↩ _m	The mask to compare against.

Returns

(M & __m) != 0.

Referenced by std::time_get< _CharT, _InIter >::get().

narrow() [1/2]

```
template<typename _CharT>
char std::__ctype_abstract_base< _CharT >::narrow (
            char_type __c,
            char __dfault) const  [inline], [inherited]
```

Narrow char_type to char.
This function converts the char_type to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. It does so by returning ctype<char_type>::do_narrow(__c).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char_type to convert.
__dfault	Char to return if conversion fails.

Returns

The converted char.

Referenced by std::time_get< _CharT, _InIter >::do_get_year(), std::time_get< _CharT, _InIter >::get(), and std::time_put< _CharT, _OutIter >::put().

narrow() [2/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::narrow (
            const char_type * __lo,
            const char_type * __hi,
            char __dfault,
            char * __to) const  [inline], [inherited]
```

Narrow array to char array.
This function converts each char_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char_type in the input that cannot be converted, *dfault* is used instead. It does so by returning ctype<char_type>::do_narrow(__lo, __hi, __dfault, __to).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

scan_is()

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::scan_is (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [inline], [inherited]
```

Find char_type matching a mask.
This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char_type>::do_scan_is().

Parameters

_↩ _m	The mask to compare against.

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to matching char_type if found, else __*hi*.

scan_not()

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::scan_not (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [inline], [inherited]
```

Find char_type not matching a mask.
This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char_type>::do_scan_not().

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to first char in range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to non-matching char if found, else __*hi*.

tolower() [1/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::tolower (
            char_type * __lo,
            const char_type * __hi) const  [inline], [inherited]
```

Convert array to lowercase.
This function converts each char_type in the range [__lo,__hi) to lowercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>:: do_tolower(__lo, __hi).

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

tolower() [2/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::tolower (
            char_type __c) const  [inline], [inherited]
```

Convert to lowercase.
This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_tolower(c).

Parameters

_↩ _c	The char_type to convert.

Returns

The lowercase char_type if convertible, else *__c*.

Referenced by std::time_get< _CharT, _InIter >::get().

toupper() [1/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::toupper (
            char_type * __lo,
            const char_type * __hi) const  [inline], [inherited]
```

Convert array to uppercase.
This function converts each char_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>:: do_toupper(lo, hi).

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

toupper() [2/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::toupper (
            char_type __c) const  [inline], [inherited]
```

Convert to uppercase.
This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_toupper().

Parameters

_↩ _c	The char_type to convert.

Returns

The uppercase char_type if convertible, else __c.

Referenced by std::time_get< _CharT, _InIter >::get().

widen() [1/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::widen (
            char __c) const  [inline], [inherited]
```

Widen char to char_type.
This function converts the char argument to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _c	The char to convert.

Returns

The converted char_type.

Referenced by std::money_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::money_put< _CharT, _Out and std::time_put< _CharT, _OutIter >::do_put().

widen() [2/2]

```
template<typename _CharT>
const char * std::__ctype_abstract_base< _CharT >::widen (
            const char * __lo,
            const char * __hi,
            char_type * __to) const  [inline], [inherited]
```

Widen array to char_type.
This function converts each char in the input to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.
_↩ _to	Pointer to the destination array.

Returns

__hi.

5.355.3 Member Data Documentation

id

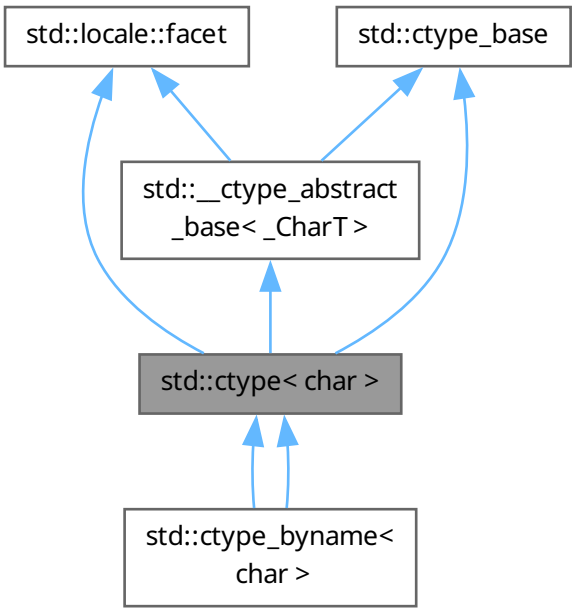
```
template<typename _CharT>
locale::id std::ctype< _CharT >::id  [static]
```

The facet id for ctype<char_type>
The documentation for this class was generated from the following file:

- locale_facets.h

5.356 std::ctype< char > Class Reference

`#include <locale_facets.h>`
Inheritance diagram for std::ctype< char >:



Public Types

- typedef const int * **__to_type**
- typedef char **char_type**
- typedef char char_type
- typedef __ctype_abstract_base< char >::mask **mask**

Public Member Functions

- ctype (__c_locale __cloc, const mask *__table=0, bool __del=false, size_t __refs=0)
- ctype (const mask *__table=0, bool __del=false, size_t __refs=0)
- **ctype** (size_t __refs=0)
- const char_type * is (const char_type *__lo, const char_type *__hi, mask *__vec) const
- bool is (mask __m, char_type __c) const
- const char * is (const char *__lo, const char *__hi, mask *__vec) const
- const char_type * is (const char_type *__lo, const char_type *__hi, mask *__vec) const
- bool is (mask __m, char __c) const
- bool is (mask __m, char_type __c) const
- char narrow (char_type __c, char __dfault) const
- const char_type * narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__to) const
- char narrow (char_type __c, char __dfault) const
- char narrow (char_type __c, char __dfault) const
- const char_type * narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__to) const
- const char_type * narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__to) const
- const char_type * scan_is (mask __m, const char_type *__lo, const char_type *__hi) const
- const char * scan_is (mask __m, const char *__lo, const char *__hi) const
- const char_type * scan_is (mask __m, const char_type *__lo, const char_type *__hi) const
- const char_type * scan_not (mask __m, const char_type *__lo, const char_type *__hi) const
- const char * scan_not (mask __m, const char *__lo, const char *__hi) const
- const char_type * scan_not (mask __m, const char_type *__lo, const char_type *__hi) const
- const mask * table () const throw ()
- const char_type * tolower (char_type *__lo, const char_type *__hi) const
- char_type tolower (char_type __c) const
- const char_type * tolower (char_type *__lo, const char_type *__hi) const
- const char_type * tolower (char_type *__lo, const char_type *__hi) const
- char_type tolower (char_type __c) const
- char_type tolower (char_type __c) const
- const char_type * toupper (char_type *__lo, const char_type *__hi) const
- char_type toupper (char_type __c) const
- const char_type * toupper (char_type *__lo, const char_type *__hi) const
- const char_type * toupper (char_type *__lo, const char_type *__hi) const
- char_type toupper (char_type __c) const
- char_type toupper (char_type __c) const
- const char * widen (const char *__lo, const char *__hi, char_type *__to) const
- char_type widen (char __c) const
- char_type widen (char __c) const
- const char * widen (const char *__lo, const char *__hi, char_type *__to) const
- const char * widen (const char *__lo, const char *__hi, char_type *__to) const

Static Public Member Functions

- static const mask * classic_table () throw ()

Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **blank**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static locale::id id
- static locale::id id
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const size_t table_size
- static const mask **upper**
- static const mask **xdigit**

Protected Member Functions

- virtual ~ctype ()
- virtual const char_type * do_is (const char_type *__lo, const char_type *__hi, mask *__vec) const =0
- virtual bool do_is (mask __m, char_type __c) const =0
- virtual const char_type * do_is (const char_type *__lo, const char_type *__hi, mask *__vec) const
- virtual bool do_is (mask __m, char_type __c) const
- virtual char do_narrow (char_type __c, char __dfault) const =0
- virtual const char_type * do_narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__to) const =0
- virtual char do_narrow (char_type __c, char __dfault) const
- virtual char do_narrow (char_type, char __dfault) const
- virtual const char_type * do_narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__to) const
- virtual const char_type * do_narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__to) const
- virtual const char_type * do_scan_is (mask __m, const char_type *__lo, const char_type *__hi) const =0
- virtual const char_type * do_scan_is (mask __m, const char_type *__lo, const char_type *__hi) const
- virtual const char_type * do_scan_not (mask __m, const char_type *__lo, const char_type *__hi) const =0
- virtual const char_type * do_scan_not (mask __m, const char_type *__lo, const char_type *__hi) const
- virtual const char_type * do_tolower (char_type *__lo, const char_type *__hi) const =0
- virtual char_type do_tolower (char_type __c) const =0
- virtual const char_type * do_tolower (char_type *__lo, const char_type *__hi) const
- virtual const char_type * do_tolower (char_type *__lo, const char_type *__hi) const
- virtual char_type do_tolower (char_type __c) const
- virtual char_type do_tolower (char_type __c) const
- virtual const char_type * do_toupper (char_type *__lo, const char_type *__hi) const =0
- virtual char_type do_toupper (char_type __c) const =0
- virtual const char_type * do_toupper (char_type *__lo, const char_type *__hi) const
- virtual const char_type * do_toupper (char_type *__lo, const char_type *__hi) const
- virtual char_type do_toupper (char_type __c) const
- virtual char_type do_toupper (char_type __c) const
- virtual const char * do_widen (const char *__lo, const char *__hi, char_type *__to) const =0
- virtual char_type do_widen (char __c) const
- virtual char_type do_widen (char __c) const
- virtual const char * do_widen (const char *__lo, const char *__hi, char_type *__dest) const
- virtual const char * do_widen (const char *__lo, const char *__hi, char_type *__to) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale **_M_c_locale_ctype**
- bool **_M_del**
- char **_M_narrow** [1+static_cast< unsigned char >(-1)]
- char **_M_narrow_ok**
- const mask * **_M_table**
- __to_type **_M_tolower**
- __to_type **_M_toupper**
- char **_M_widen** [1+static_cast< unsigned char >(-1)]
- char **_M_widen_ok**

5.356.1 Detailed Description

The ctype<char> specialization.
This class defines classification and conversion functions for the char type. It gets used by char streams for many I/O operations. The char specialization provides a number of optimizations as well.

5.356.2 Member Typedef Documentation

char_type

```
typedef char std::ctype< char >::char_type
```

Typedef for the template parameter char.

5.356.3 Constructor & Destructor Documentation

ctype() [1/2]

```
std::ctype< char >::ctype (
            const mask * __table = 0,
            bool __del = false,
            size_t __refs = 0)  [explicit]
```

Constructor performs initialization.
This is the constructor provided by the standard.

Parameters

__table	If non-zero, table is used as the per-char mask. Else classic_table() is used.
__del	If true, passes ownership of table to this facet.
__refs	Passed to the base facet class.

Referenced by ~ctype().

ctype() [2/2]

```
std::ctype< char >::ctype (
            __c_locale __cloc,
            const mask * __table = 0,
            bool __del = false,
            size_t __refs = 0)  [explicit]
```

Constructor performs static initialization.
This constructor is used to construct the initial C locale facet.

Parameters

__cloc	Handle to C locale data.
__table	If non-zero, table is used as the per-char mask.
__del	If true, passes ownership of table to this facet.
__refs	Passed to the base facet class.

~ctype()

```
virtual std::ctype< char >::~ctype ()  [protected], [virtual]
```

Destructor.
This function deletes table() if *del* was true in the constructor.
References ctype().

5.356.4 Member Function Documentation

classic_table()

```
static const mask * std::ctype< char >::classic_table () throw ( )   [static]
```

Returns a pointer to the C locale mask table.

do_is() [1/4]

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base< _CharT >::do_is (
            const char_type * __lo,
            const char_type * __hi,
            mask * __vec) const  [protected], [pure virtual], [inherited]
```

Return a mask array.
This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.
do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__vec	Pointer to an array of mask storage.

Returns

__hi.

Implemented in std::ctype< _CharT >, std::ctype< wchar_t >, std::ctype< wchar_t >, std::mask< _CharT >, and std::mask< _CharT >.

do_is() [2/4]

```
template<typename _CharT>
virtual bool std::__ctype_abstract_base< _CharT >::do_is (
            mask __m,
            char_type __c) const  [protected], [pure virtual], [inherited]
```

Test char_type classification.
This function finds a mask M for *c* and compares it to mask *m*.
do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

__c	The char_type to find the mask of.
__m	The mask to compare against.

Returns

(M & __m) != 0.

Implemented in std::ctype< _CharT >, std::ctype< wchar_t >, std::ctype< wchar_t >, std::mask< _CharT >, and std::mask< _CharT >.
Referenced by std::mask< _CharT >::is(), and std::mask< _CharT >::is().

do_is() [3/4]

```
virtual const char_type * std::ctype< char >::do_is (
            const char_type * __lo,
            const char_type * __hi,
            mask * __vec) const  [protected], [virtual]
```

Return a mask array.
This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.
do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__vec	Pointer to an array of mask storage.

Returns

__hi.

do_is() [4/4]

```
virtual bool std::ctype< char >::do_is (
            mask __m,
            char_type __c) const  [protected], [virtual]
```

Test char_type classification.
This function finds a mask M for *c* and compares it to mask *m*.
do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

_↩ _c	The char_type to find the mask of.
_↩ _m	The mask to compare against.

Returns

(M & __m) != 0.

do_narrow() [1/6]

```
template<typename _CharT>
virtual char std::__ctype_abstract_base< _CharT >::do_narrow (
            char_type __c,
            char __dfault) const  [protected], [pure virtual], [inherited]
```

Narrow char_type to char.
This virtual function converts the argument to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead.
do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char_type to convert.
__dfault	Char to return if conversion fails.

Returns

The converted char.

Implemented in std::ctype< _CharT >, std::ctype< wchar_t >, std::ctype< wchar_t >, std::mask< _CharT >, and std::mask< _CharT >.
Referenced by std::mask< _CharT >::narrow(), and std::mask< _CharT >::narrow().

do_narrow() [2/6]

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base< _CharT >::do_narrow (
            const char_type * __lo,
            const char_type * __hi,
```

```
            char __dfault,
            char * __to) const  [protected], [pure virtual], [inherited]
```

Narrow char_type array to char.
This virtual function converts each char_type in the range [__lo,__hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, *__dfault* is used instead.
do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

Implemented in std::ctype< _CharT >, std::ctype< wchar_t >, std::ctype< wchar_t >, std::mask< _CharT >, and std::mask< _CharT >.

do_narrow() [3/6]

```
virtual char std::ctype< char >::do_narrow (
            char_type __c,
            char __dfault) const  [inline], [protected], [virtual]
```

Narrow char.
This virtual function converts the char to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. For an underived ctype<char> facet, *c* will be returned unchanged.
do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char to convert.
__dfault	Char to return if conversion fails.

Returns

The converted char.

do_narrow() [4/6]

```
virtual char std::ctype< char >::do_narrow (
            char_type ,
            char __dfault) const  [protected], [virtual]
```

Narrow char_type to char.
This virtual function converts the argument to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead.

do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char_type to convert.
__dfault	Char to return if conversion fails.

Returns

The converted char.

do_narrow() [5/6]

```
virtual const char_type * std::ctype< char >::do_narrow (
            const char_type * __lo,
            const char_type * __hi,
            char __dfault,
            char * __to) const  [protected], [virtual]
```

Narrow char_type array to char.

This virtual function converts each char_type in the range [__lo,__hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, *__dfault* is used instead.

do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

do_narrow() [6/6]

```
virtual const char_type * std::ctype< char >::do_narrow (
            const char_type * __lo,
            const char_type * __hi,
            char __dfault,
            char * __to) const  [inline], [protected], [virtual]
```

Narrow char array to char array.

This virtual function converts each char in the range [lo,hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *dfault* is used instead. For an underived ctype<char> facet, the argument will be copied unchanged.

do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

do_scan_is() [1/2]

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base< _CharT >::do_scan_is (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [protected], [pure virtual], [inherited]
```

Find char_type matching mask.
This function searches for and returns the first char_type c in [__lo,__hi) for which is(__m,c) is true.
do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to a matching char_type if found, else *__hi*.

Implemented in std::ctype< _CharT >, std::ctype< wchar_t >, std::ctype< wchar_t >, std::mask< _CharT >, and std::mask< _CharT >.
Referenced by std::mask< _CharT >::scan_is().

do_scan_is() [2/2]

```
virtual const char_type * std::ctype< char >::do_scan_is (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [protected], [virtual]
```

Find char_type matching mask.
This function searches for and returns the first char_type c in [__lo,__hi) for which is(__m,c) is true.
do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to a matching char_type if found, else __hi.

do_scan_not() [1/2]

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base< _CharT >::do_scan_not (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [protected], [pure virtual], [inherited]
```

Find char_type not matching mask.
This function searches for and returns a pointer to the first char_type c of [lo,hi) for which is(m,c) is false.
do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to a non-matching char_type if found, else __hi.

Implemented in std::ctype< _CharT >, std::ctype< wchar_t >, std::ctype< wchar_t >, std::mask< _CharT >, and std::mask< _CharT >.
Referenced by std::mask< _CharT >::scan_not().

do_scan_not() [2/2]

```
virtual const char_type * std::ctype< char >::do_scan_not (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [protected], [virtual]
```

Find char_type not matching mask.
This function searches for and returns a pointer to the first char_type c of [lo,hi) for which is(m,c) is false.
do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to a non-matching char_type if found, else __hi.

do_tolower() [1/6]

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base< _CharT >::do_tolower (
            char_type * __lo,
            const char_type * __hi) const  [protected], [pure virtual], [inherited]
```

Convert array to lowercase.
This virtual function converts each char_type in the range [__lo,__hi) to lowercase if possible. Other elements remain untouched.
do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

Implemented in std::ctype< _CharT >, std::ctype< wchar_t >, std::ctype< wchar_t >, std::mask< _CharT >, and std::mask< _CharT >.

do_tolower() [2/6]

```
template<typename _CharT>
virtual char_type std::__ctype_abstract_base< _CharT >::do_tolower (
            char_type __c) const  [protected], [pure virtual], [inherited]
```

Convert to lowercase.
This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.
do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

_↩ _c	The char_type to convert.

Returns

The lowercase char_type if convertible, else *__c*.

Implemented in std::ctype< _CharT >, std::ctype< wchar_t >, std::ctype< wchar_t >, std::mask< _CharT >, and std::mask< _CharT >.
Referenced by std::mask< _CharT >::tolower(), and std::mask< _CharT >::tolower().

do_tolower() [3/6]

```
virtual const char_type * std::ctype< char >::do_tolower (
            char_type * __lo,
            const char_type * __hi) const  [protected], [virtual]
```

Convert array to lowercase.
This virtual function converts each char_type in the range [__lo,__hi) to lowercase if possible. Other elements remain untouched.
do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

__hi.

do_tolower() [4/6]

```
virtual const char_type * std::ctype< char >::do_tolower (
            char_type * __lo,
            const char_type * __hi) const  [protected], [virtual]
```

Convert array to lowercase.
This virtual function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.
do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

__lo	Pointer to first char in range.
__hi	Pointer to end of range.

Returns

__hi.

References do_tolower().

do_tolower() [5/6]

```
virtual char_type std::ctype< char >::do_tolower (
            char_type __c) const  [protected], [virtual]
```

Convert to lowercase.
This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.
do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

_↩ _c	The char_type to convert.

Returns

The lowercase char_type if convertible, else __c.

do_tolower() [6/6]

```
virtual char_type std::ctype< char >::do_tolower (
            char_type __c) const  [protected], [virtual]
```

Convert to lowercase.
This virtual function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.
do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

_↩ _c	The char to convert.

Returns

The lowercase char if convertible, else __c.

References do_tolower().
Referenced by do_tolower(), and do_tolower().

do_toupper() [1/6]

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base< _CharT >::do_toupper (
            char_type * __lo,
            const char_type * __hi) const  [protected], [pure virtual], [inherited]
```

Convert array to uppercase.
This virtual function converts each char_type in the range [__lo,__hi) to uppercase if possible. Other elements remain untouched.
do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

Implemented in std::ctype< _CharT >, std::ctype< wchar_t >, std::ctype< wchar_t >, std::mask< _CharT >, and std::mask< _CharT >.

do_toupper() [2/6]

```
template<typename _CharT>
virtual char_type std::__ctype_abstract_base< _CharT >::do_toupper (
            char_type __c) const  [protected], [pure virtual], [inherited]
```

Convert to uppercase.
This virtual function converts the char_type argument to uppercase if possible. If not possible (for example, '2'), returns the argument.
do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

_↩ _c	The char_type to convert.

Returns

The uppercase char_type if convertible, else *__c*.

Implemented in std::ctype< _CharT >, std::ctype< wchar_t >, std::ctype< wchar_t >, std::mask< _CharT >, and std::mask< _CharT >.
Referenced by std::mask< _CharT >::toupper(), and std::mask< _CharT >::toupper().

do_toupper() [3/6]

```
virtual const char_type * std::ctype< char >::do_toupper (
            char_type * __lo,
            const char_type * __hi) const  [protected], [virtual]
```

Convert array to uppercase.
This virtual function converts each char_type in the range [__lo,__hi) to uppercase if possible. Other elements remain untouched.
do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

do_toupper() [4/6]

```
virtual const char_type * std::ctype< char >::do_toupper (
            char_type * __lo,
            const char_type * __hi) const  [protected], [virtual]
```

Convert array to uppercase.
This virtual function converts each char in the range [lo,hi) to uppercase if possible. Other chars remain untouched.
do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

__hi.

References do_toupper().

do_toupper() [5/6]

```
virtual char_type std::ctype< char >::do_toupper (
            char_type __c) const  [protected], [virtual]
```

Convert to uppercase.
This virtual function converts the char_type argument to uppercase if possible. If not possible (for example, '2'), returns the argument.
do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

__c	The char_type to convert.

Returns

The uppercase char_type if convertible, else *__c*.

do_toupper() [6/6]

```
virtual char_type std::ctype< char >::do_toupper (
            char_type __c) const  [protected], [virtual]
```

Convert to uppercase.
This virtual function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.
do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

_↩ _c	The char to convert.

Returns

The uppercase char if convertible, else __c.

References do_toupper().
Referenced by do_toupper(), and do_toupper().

do_widen() [1/5]

```
template<typename _CharT>
virtual const char * std::__ctype_abstract_base< _CharT >::do_widen (
            const char * __lo,
            const char * __hi,
            char_type * __to) const  [protected], [pure virtual], [inherited]
```

Widen char array.
This function converts each char in the input to char_type using the simplest reasonable transformation.
do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _lo	Pointer to start range.
_↩ _hi	Pointer to end of range.
_↩ _to	Pointer to the destination array.

Returns

__hi.

Implemented in std::ctype< _CharT >, std::ctype< wchar_t >, std::ctype< wchar_t >, std::mask< _CharT >, and std::mask< _CharT >.

do_widen() [2/5]

```
virtual char_type std::ctype< char >::do_widen (
            char __c) const  [protected], [virtual]
```

Widen char.
This virtual function converts the char to char_type using the simplest reasonable transformation.
do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _c	The char to convert.

Returns

The converted char_type

Implements std::__ctype_abstract_base< _CharT >.

do_widen() [3/5]

```
virtual char_type std::ctype< char >::do_widen (
            char __c) const  [inline], [protected], [virtual]
```

Widen char.
This virtual function converts the char to char using the simplest reasonable transformation. For an underived ctype<char> facet, the argument will be returned unchanged.
do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _c	The char to convert.

Returns

The converted character.

Implements std::__ctype_abstract_base< _CharT >.
References do_widen().
Referenced by do_widen().

do_widen() [4/5]

```
virtual const char * std::ctype< char >::do_widen (
            const char * __lo,
            const char * __hi,
            char_type * __dest) const  [protected], [virtual]
```

Widen char array.
This function converts each char in the input to char_type using the simplest reasonable transformation.
do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _lo	Pointer to start range.
_↩ _hi	Pointer to end of range.
_↩ _to	Pointer to the destination array.

Returns

__hi.

do_widen() [5/5]

```
virtual const char * std::ctype< char >::do_widen (
            const char * __lo,
            const char * __hi,
            char_type * __to) const  [inline], [protected], [virtual]
```

Widen char array.

This function converts each char in the range [lo,hi) to char using the simplest reasonable transformation. For an underived ctype<char> facet, the argument will be copied unchanged.

do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.
_↩ _to	Pointer to the destination array.

Returns

__hi.

is() [1/6]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::is (
            const char_type * __lo,
            const char_type * __hi,
            mask * __vec) const  [inline], [inherited]
```

Return a mask array.

This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char_type>::do_is().

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__vec	Pointer to an array of mask storage.

Returns

__hi.

is() [2/6]

```
template<typename _CharT>
bool std::__ctype_abstract_base< _CharT >::is (
            mask __m,
            char_type __c) const  [inline], [inherited]
```

Test char_type classification.
This function finds a mask M for __*c* and compares it to mask __*m*. It does so by returning the value of ctype<char_↩ type>::do_is().

Parameters

_↩ _*c*	The char_type to compare the mask of.
_↩ _*m*	The mask to compare against.

Returns

(M & __m) != 0.

Referenced by std::time_get< _CharT, _InIter >::get().

is() [3/6]

```
const char * std::ctype< char >::is (
             const char * __lo,
             const char * __hi,
             mask * __vec) const  [inline]
```

Return a mask array.
This function finds the mask for each char in the range [lo, hi) and successively writes it to vec. vec must have as many elements as the char array.

Parameters

__*lo*	Pointer to start of range.
__*hi*	Pointer to end of range.
__*vec*	Pointer to an array of mask storage.

Returns

__*hi*.

is() [4/6]

```
const char_type * std::__ctype_abstract_base< char >::is (
             const char_type * __lo,
             const char_type * __hi,
             mask * __vec) const  [inline]
```

Return a mask array.
This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char_type>::do_is().

Parameters

__*lo*	Pointer to start of range.
__*hi*	Pointer to end of range.
__*vec*	Pointer to an array of mask storage.

Returns

__hi.

is() [5/6]

```
bool std::ctype< char >::is (
            mask __m,
            char __c) const  [inline]
```

Test char classification.
This function compares the mask table[c] to *__m*.

Parameters

_↩ _*c*	The char to compare the mask of.
_↩ _*m*	The mask to compare against.

Returns

True if __m & table[__c] is true, false otherwise.

is() [6/6]

```
bool std::__ctype_abstract_base< char >::is (
            mask __m,
            char_type __c) const  [inline]
```

Test char_type classification.
This function finds a mask M for *__c* and compares it to mask *__m*. It does so by returning the value of ctype<char_↩ type>::do_is().

Parameters

_↩ _*c*	The char_type to compare the mask of.
_↩ _*m*	The mask to compare against.

Returns

(M & __m) != 0.

narrow() [1/6]

```
template<typename _CharT>
char std::__ctype_abstract_base< _CharT >::narrow (
            char_type __c,
            char __dfault) const  [inline], [inherited]
```

Narrow char_type to char.
This function converts the char_type to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. It does so by returning ctype<char_type>::do_narrow(__c).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char_type to convert.
__dfault	Char to return if conversion fails.

Returns

The converted char.

Referenced by std::time_get< _CharT, _InIter >::do_get_year(), std::time_get< _CharT, _InIter >::get(), and std::time_put< _CharT, _OutIter >::put().

narrow() [2/6]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::narrow (
            const char_type * __lo,
            const char_type * __hi,
            char __dfault,
            char * __to) const  [inline], [inherited]
```

Narrow array to char array.
This function converts each char_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char_type in the input that cannot be converted, *dfault* is used instead. It does so by returning ctype<char_type>::do_narrow(__lo, __hi, __dfault, __to).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

narrow() [3/6]

```
char std::__ctype_abstract_base< char >::narrow (
            char_type __c,
            char __dfault) const  [inline]
```

Narrow char_type to char.
This function converts the char_type to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. It does so by returning ctype<char_type>::do_narrow(__c).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char_type to convert.
__dfault	Char to return if conversion fails.

Returns

The converted char.

narrow() [4/6]

```
char std::ctype< char >::narrow (
            char_type __c,
            char __dfault) const  [inline]
```

Narrow char.
This function converts the char to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. For an underived ctype<char> facet, *c* will be returned unchanged.
This function works as if it returns ctype<char>::do_narrow(c). do_narrow() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char to convert.
__dfault	Char to return if conversion fails.

Returns

The converted character.

References std::ctype< _CharT >::do_narrow().

narrow() [5/6]

```
const char_type * std::__ctype_abstract_base< char >::narrow (
            const char_type * __lo,
            const char_type * __hi,
            char __dfault,
            char * __to) const  [inline]
```

Narrow array to char array.
This function converts each char_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char_type in the input that cannot be converted, *dfault* is used instead. It does so by returning ctype<char_type>::do_narrow(__lo, __hi, __dfault, __to).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

narrow() [6/6]

```
const char_type * std::ctype< char >::narrow (
            const char_type * __lo,
            const char_type * __hi,
            char __dfault,
            char * __to) const  [inline]
```

Narrow char array.
This function converts each char in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *dfault* is used instead. For an underived ctype<char> facet, the argument will be copied unchanged.
This function works as if it returns ctype<char>::do_narrow(lo, hi, dfault, to). do_narrow() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

References std::ctype< _CharT >::do_narrow().

scan_is() [1/3]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::scan_is (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [inline], [inherited]
```

Find char_type matching a mask.
This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char_type>::do_scan_is().

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to matching char_type if found, else *__hi*.

scan_is() [2/3]

```
const char * std::ctype< char >::scan_is (
            mask __m,
            const char * __lo,
            const char * __hi) const  [inline]
```

Find char matching a mask.
This function searches for and returns the first char in [lo,hi) for which is(m,char) is true.

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to a matching char if found, else *__hi*.

scan_is() [3/3]

```
const char_type * std::__ctype_abstract_base< char >::scan_is (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [inline]
```

Find char_type matching a mask.
This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char_type>::do_scan_is().

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to matching char_type if found, else *__hi*.

scan_not() [1/3]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::scan_not (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [inline], [inherited]
```

Find char_type not matching a mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char_type>::do_scan_not().

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to first char in range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to non-matching char if found, else __hi.

scan_not() [2/3]

```
const char * std::ctype< char >::scan_not (
            mask __m,
            const char * __lo,
            const char * __hi) const  [inline]
```

Find char not matching a mask.
This function searches for and returns a pointer to the first char in [__lo,__hi) for which is(m,char) is false.

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to a non-matching char if found, else __hi.

scan_not() [3/3]

```
const char_type * std::__ctype_abstract_base< char >::scan_not (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [inline]
```

Find char_type not matching a mask.
This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char_type>::do_scan_not().

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to first char in range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to non-matching char if found, else __*hi*.

table()

```
const mask * std::ctype< char >::table () const throw ( )   [inline]
```

Returns a pointer to the mask table provided to the constructor, or the default from classic_table() if none was provided.

tolower() [1/6]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::tolower (
            char_type * __lo,
            const char_type * __hi) const  [inline], [inherited]
```

Convert array to lowercase.

This function converts each char_type in the range [__lo,__hi) to lowercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>:: do_tolower(__lo, __hi).

Parameters

_↩ _*lo*	Pointer to start of range.
_↩ _*hi*	Pointer to end of range.

Returns

__*hi*.

tolower() [2/6]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::tolower (
            char_type __c) const  [inline], [inherited]
```

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_tolower(c).

Parameters

_↩ _*c*	The char_type to convert.

Returns

The lowercase char_type if convertible, else __*c*.

Referenced by std::time_get< _CharT, _InIter >::get().

tolower() [3/6]

```
const char_type * std::__ctype_abstract_base< char >::tolower (
            char_type * __lo,
            const char_type * __hi) const  [inline]
```

Convert array to lowercase.
This function converts each char_type in the range [__lo,__hi) to lowercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>:: do_tolower(__lo, __hi).

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

tolower() [4/6]

```
const char_type * std::ctype< char >::tolower (
            char_type * __lo,
            const char_type * __hi) const  [inline]
```

Convert array to lowercase.
This function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.
tolower() acts as if it returns ctype<char>:: do_tolower(__lo, __hi). do_tolower() must always return the same result for the same input.

Parameters

_↩ _lo	Pointer to first char in range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

References std::ctype< _CharT >::do_tolower().

tolower() [5/6]

```
char_type std::__ctype_abstract_base< char >::tolower (
            char_type __c) const  [inline]
```

Convert to lowercase.
This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_tolower(c).

Parameters

_↩ _c	The char_type to convert.

Returns

The lowercase char_type if convertible, else __c.

tolower() [6/6]

```
char_type std::ctype< char >::tolower (
            char_type __c) const  [inline]
```

Convert to lowercase.
This function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument. tolower() acts as if it returns ctype<char>::do_tolower(__c). do_tolower() must always return the same result for the same input.

Parameters

_↩ _c	The char to convert.

Returns

The lowercase char if convertible, else __c.

References std::ctype< _CharT >::do_tolower().

toupper() [1/6]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::toupper (
            char_type * __lo,
            const char_type * __hi) const  [inline], [inherited]
```

Convert array to uppercase.
This function converts each char_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>:: do_toupper(lo, hi).

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

toupper() [2/6]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::toupper (
            char_type __c) const  [inline], [inherited]
```

Convert to uppercase.
This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_toupper().

Parameters

_↩ _c	The char_type to convert.

Returns

The uppercase char_type if convertible, else __c.

Referenced by std::time_get< _CharT, _InIter >::get().

toupper() [3/6]

```
const char_type * std::__ctype_abstract_base< char >::toupper (
            char_type * __lo,
            const char_type * __hi) const  [inline]
```

Convert array to uppercase.
This function converts each char_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>:: do_toupper(lo, hi).

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

toupper() [4/6]

```
const char_type * std::ctype< char >::toupper (
            char_type * __lo,
            const char_type * __hi) const  [inline]
```

Convert array to uppercase.
This function converts each char in the range [__lo,__hi) to uppercase if possible. Other chars remain untouched. toupper() acts as if it returns ctype<char>:: do_toupper(__lo, __hi). do_toupper() must always return the same result for the same input.

Parameters

_↩ _lo	Pointer to first char in range.

_↩ _hi	Pointer to end of range.

Returns

__hi.

References std::ctype< _CharT >::do_toupper().

toupper() [5/6]

```
char_type std::__ctype_abstract_base< char >::toupper (
            char_type __c) const  [inline]
```

Convert to uppercase.
This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_toupper().

Parameters

_↩ _c	The char_type to convert.

Returns

The uppercase char_type if convertible, else *__c*.

toupper() [6/6]

```
char_type std::ctype< char >::toupper (
            char_type __c) const  [inline]
```

Convert to uppercase.
This function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument. toupper() acts as if it returns ctype<char>::do_toupper(c). do_toupper() must always return the same result for the same input.

Parameters

_↩ _c	The char to convert.

Returns

The uppercase char if convertible, else *__c*.

References std::ctype< _CharT >::do_toupper().

widen() [1/5]

```
template<typename _CharT>
const char * std::__ctype_abstract_base< _CharT >::widen (
            const char * __lo,
            const char * __hi,
            char_type * __to) const  [inline], [inherited]
```

Widen array to char_type.
This function converts each char in the input to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _*lo*	Pointer to start of range.
_↩ _*hi*	Pointer to end of range.
_↩ _*to*	Pointer to the destination array.

Returns

__hi.

widen() [2/5]

```
char_type std::__ctype_abstract_base< char >::widen (
            char __c) const  [inline]
```

Widen char to char_type.
This function converts the char argument to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _*c*	The char to convert.

Returns

The converted char_type.

widen() [3/5]

```
char_type std::ctype< char >::widen (
            char __c) const  [inline]
```

Widen char.
This function converts the char to char_type using the simplest reasonable transformation. For an underived ctype<char> facet, the argument will be returned unchanged.
This function works as if it returns ctype<char>::do_widen(c). do_widen() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _*c*	The char to convert.

Returns

The converted character.

References std::ctype< _CharT >::do_widen().

widen() [4/5]

```
const char * std::__ctype_abstract_base< char >::widen (
            const char * __lo,
            const char * __hi,
            char_type * __to) const  [inline]
```

Widen array to char_type.
This function converts each char in the input to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__to	Pointer to the destination array.

Returns

__hi.

widen() [5/5]

```
const char * std::ctype< char >::widen (
            const char * __lo,
            const char * __hi,
            char_type * __to) const  [inline]
```

Widen char array.
This function converts each char in the input to char using the simplest reasonable transformation. For an underived ctype<char> facet, the argument will be copied unchanged.
This function works as if it returns ctype<char>::do_widen(c). do_widen() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to first char in range.
__hi	Pointer to end of range.
__to	Pointer to the destination array.

Returns

__hi.

References std::ctype< _CharT >::do_widen().

5.356.5 Member Data Documentation

id [1/2]

```
locale::id std::ctype< char >::id  [static]
```
The facet id for ctype<char_type>

id [2/2]

```
locale::id std::ctype< char >::id  [static]
```
The facet id for ctype<char>

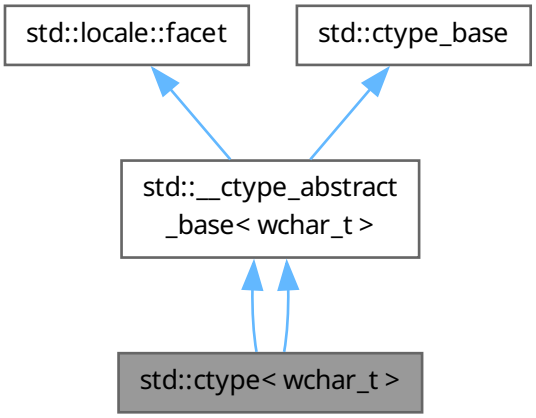
table_size

```
const size_t std::ctype< char >::table_size  [static]
```
The size of the mask table. It is SCHAR_MAX + 1.
The documentation for this class was generated from the following files:

- locale_facets.h
- ctype_inline.h

5.357 std::ctype< wchar_t > Class Reference

```
#include <locale_facets.h>
```
Inheritance diagram for std::ctype< wchar_t >:



Public Types

- typedef const int * **__to_type**
- typedef wctype_t **__wmask_type**
- typedef wchar_t **char_type**
- typedef wchar_t char_type
- typedef __ctype_abstract_base< wchar_t >::mask **mask**

Public Member Functions

- ctype (__c_locale __cloc, size_t __refs=0)
- **ctype** (size_t __refs=0)
- ctype (size_t __refs=0)
- const char_type * is (const char_type *__lo, const char_type *__hi, mask *__vec) const
- bool is (mask __m, char_type __c) const
- char narrow (char_type __c, char __dfault) const
- const char_type * narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__to) const
- const char_type * scan_is (mask __m, const char_type *__lo, const char_type *__hi) const
- const char_type * scan_not (mask __m, const char_type *__lo, const char_type *__hi) const
- const char_type * tolower (char_type *__lo, const char_type *__hi) const
- char_type tolower (char_type __c) const
- const char_type * toupper (char_type *__lo, const char_type *__hi) const
- char_type toupper (char_type __c) const
- char_type widen (char __c) const
- const char * widen (const char *__lo, const char *__hi, char_type *__to) const

Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **blank**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static locale::id id
- static locale::id id
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

Protected Member Functions

- virtual ~ctype ()
- __wmask_type **_M_convert_to_wmask** (const mask __m) const throw ()
- void **_M_initialize_ctype** () throw ()
- virtual const char_type * do_is (const char_type *__lo, const char_type *__hi, mask *__vec) const
- virtual const char_type * do_is (const char_type *__lo, const char_type *__hi, mask *__vec) const
- virtual bool do_is (mask __m, char_type __c) const
- virtual bool do_is (mask __m, char_type __c) const
- virtual char do_narrow (char_type __c, char __dfault) const
- virtual char do_narrow (char_type, char __dfault) const
- virtual const char_type * do_narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__to) const
- virtual const char_type * do_narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__to) const
- virtual const char_type * do_scan_is (mask __m, const char_type *__lo, const char_type *__hi) const
- virtual const char_type * do_scan_is (mask __m, const char_type *__lo, const char_type *__hi) const

- virtual const char_type * do_scan_not (mask __m, const char_type *__lo, const char_type *__hi) const
- virtual const char_type * do_scan_not (mask __m, const char_type *__lo, const char_type *__hi) const
- virtual const char_type * do_tolower (char_type *__lo, const char_type *__hi) const
- virtual const char_type * do_tolower (char_type *__lo, const char_type *__hi) const
- virtual char_type do_tolower (char_type __c) const
- virtual char_type do_tolower (char_type __c) const
- virtual const char_type * do_toupper (char_type *__lo, const char_type *__hi) const
- virtual const char_type * do_toupper (char_type *__lo, const char_type *__hi) const
- virtual char_type do_toupper (char_type __c) const
- virtual char_type do_toupper (char_type __c) const
- virtual char_type do_widen (char __c) const
- virtual char_type do_widen (char __c) const
- virtual const char * do_widen (const char *__lo, const char *__hi, char_type *__dest) const
- virtual const char * do_widen (const char *__lo, const char *__hi, char_type *__to) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- mask **_M_bit** [16]
- __c_locale **_M_c_locale_ctype**
- char **_M_narrow** [128]
- bool **_M_narrow_ok**
- wint_t **_M_widen** [1+static_cast< unsigned char >(-1)]
- __wmask_type **_M_wmask** [16]

5.357.1 Detailed Description

The ctype<wchar_t> specialization.
This class defines classification and conversion functions for the wchar_t type. It gets used by wchar_t streams for many I/O operations. The wchar_t specialization provides a number of optimizations as well.
ctype<wchar_t> inherits its public methods from __ctype_abstract_base<wchar_t>.

5.357.2 Member Typedef Documentation

char_type

```
typedef wchar_t std::ctype< wchar_t >::char_type
```

Typedef for the template parameter wchar_t.

5.357.3 Constructor & Destructor Documentation

ctype() [1/2]

```
std::ctype< wchar_t >::ctype (
            size_t __refs = 0)  [explicit]
```

Constructor performs initialization.
This is the constructor provided by the standard.

Parameters

__refs	Passed to the base facet class.

ctype() [2/2]

```
std::ctype< wchar_t >::ctype (
            __c_locale __cloc,
            size_t __refs = 0)  [explicit]
```

Constructor performs static initialization.
This constructor is used to construct the initial C locale facet.

Parameters

__cloc	Handle to C locale data.
__refs	Passed to the base facet class.

~ctype()

```
virtual std::ctype< wchar_t >::~ctype ()  [protected], [virtual]
```

Destructor.

5.357.4 Member Function Documentation

do_is() [1/4]

```
virtual const char_type * std::ctype< wchar_t >::do_is (
            const char_type * __lo,
            const char_type * __hi,
            mask * __vec) const  [protected], [virtual]
```

Return a mask array.
This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.
do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__vec	Pointer to an array of mask storage.

Returns

__hi.

Implements std::__ctype_abstract_base< wchar_t >.

do_is() [2/4]

```
virtual const char_type * std::ctype< wchar_t >::do_is (
            const char_type * __lo,
            const char_type * __hi,
            mask * __vec) const  [protected], [virtual]
```

Return a mask array.
This function finds the mask for each wchar_t in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.
do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__vec	Pointer to an array of mask storage.

Returns

__hi.

Implements std::__ctype_abstract_base< wchar_t >.

do_is() [3/4]

```
virtual bool std::ctype< wchar_t >::do_is (
            mask __m,
            char_type __c) const  [protected], [virtual]
```

Test char_type classification.
This function finds a mask M for *c* and compares it to mask *m*.
do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

__c	The char_type to find the mask of.
__m	The mask to compare against.

Returns

(M & __m) != 0.

Implements std::__ctype_abstract_base< wchar_t >.

do_is() [4/4]

```
virtual bool std::ctype< wchar_t >::do_is (
            mask __m,
            char_type __c) const  [protected], [virtual]
```

Test wchar_t classification.
This function finds a mask M for *c* and compares it to mask *m*.
do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

_↩ _c	The wchar_t to find the mask of.
_↩ _m	The mask to compare against.

Returns

(M & __m) != 0.

Implements std::__ctype_abstract_base< wchar_t >.

do_narrow() [1/4]

```
virtual char std::ctype< wchar_t >::do_narrow (
            char_type __c,
            char __dfault) const  [protected], [virtual]
```

Narrow wchar_t to char.
This virtual function converts the argument to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. For an underived ctype<wchar_t> facet, *c* will be cast to char and returned.
do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The wchar_t to convert.
__dfault	Char to return if conversion fails.

Returns

The converted char.

Implements std::__ctype_abstract_base< wchar_t >.

do_narrow() [2/4]

```
virtual char std::ctype< wchar_t >::do_narrow (
            char_type ,
            char __dfault) const  [protected], [virtual]
```

Narrow char_type to char.
This virtual function converts the argument to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead.
do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char_type to convert.
__dfault	Char to return if conversion fails.

Returns

The converted char.

Implements std::__ctype_abstract_base< wchar_t >.

do_narrow() [3/4]

```
virtual const char_type * std::ctype< wchar_t >::do_narrow (
            const char_type * __lo,
            const char_type * __hi,
            char __dfault,
            char * __to) const  [protected], [virtual]
```

Narrow char_type array to char.
This virtual function converts each char_type in the range [__lo,__hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, *__dfault* is used instead.
do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

Implements std::__ctype_abstract_base< wchar_t >.

do_narrow() [4/4]

```
virtual const char_type * std::ctype< wchar_t >::do_narrow (
            const char_type * __lo,
            const char_type * __hi,
            char __dfault,
            char * __to) const  [protected], [virtual]
```

Narrow wchar_t array to char array.
This virtual function converts each wchar_t in the range [lo,hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any wchar_t in the input that cannot be converted, *dfault* is used instead.
For an underived ctype<wchar_t> facet, the argument will be copied, casting each element to char.
do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

Implements std::__ctype_abstract_base< wchar_t >.

do_scan_is() [1/2]

```
virtual const char_type * std::ctype< wchar_t >::do_scan_is (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [protected], [virtual]
```

Find char_type matching mask.
This function searches for and returns the first char_type c in [__lo,__hi) for which is(__m,c) is true.
do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

_↩ _*m*	The mask to compare against.
_↩ _*lo*	Pointer to start of range.
_↩ _*hi*	Pointer to end of range.

Returns

Pointer to a matching char_type if found, else __*hi*.

Implements std::__ctype_abstract_base< wchar_t >.

do_scan_is() [2/2]

```
virtual const char_type * std::ctype< wchar_t >::do_scan_is (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [protected], [virtual]
```

Find wchar_t matching mask.
This function searches for and returns the first wchar_t c in [__lo,__hi) for which is(__m,c) is true.
do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

_↩ _*m*	The mask to compare against.
_↩ _*lo*	Pointer to start of range.
_↩ _*hi*	Pointer to end of range.

Returns

Pointer to a matching wchar_t if found, else __*hi*.

Implements std::__ctype_abstract_base< wchar_t >.

do_scan_not() [1/2]

```
virtual const char_type * std::ctype< wchar_t >::do_scan_not (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [protected], [virtual]
```

Find char_type not matching mask.
This function searches for and returns a pointer to the first char_type c of [lo,hi) for which is(m,c) is false.
do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

_↩ _*m*	The mask to compare against.
_↩ _*lo*	Pointer to start of range.
_↩ _*hi*	Pointer to end of range.

Returns

Pointer to a non-matching char_type if found, else __*hi*.

Implements std::__ctype_abstract_base< wchar_t >.

do_scan_not() [2/2]

```
virtual const char_type * std::ctype< wchar_t >::do_scan_not (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [protected], [virtual]
```

Find wchar_t not matching mask.
This function searches for and returns a pointer to the first wchar_t c of [__lo,__hi) for which is(__m,c) is false.
do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

_↩ _*m*	The mask to compare against.
_↩ _*lo*	Pointer to start of range.
_↩ _*hi*	Pointer to end of range.

Returns

Pointer to a non-matching wchar_t if found, else __*hi*.

Implements std::__ctype_abstract_base< wchar_t >.

do_tolower() [1/4]

```
virtual const char_type * std::ctype< wchar_t >::do_tolower (
            char_type * __lo,
            const char_type * __hi) const  [protected], [virtual]
```

Convert array to lowercase.
This virtual function converts each char_type in the range [__lo,__hi) to lowercase if possible. Other elements remain untouched.
do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

__hi.

Implements std::__ctype_abstract_base< wchar_t >.

do_tolower() [2/4]

```
virtual const char_type * std::ctype< wchar_t >::do_tolower (
            char_type * __lo,
            const char_type * __hi) const  [protected], [virtual]
```

Convert array to lowercase.
This virtual function converts each wchar_t in the range [lo,hi) to lowercase if possible. Other elements remain untouched.
do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

__hi.

Implements std::__ctype_abstract_base< wchar_t >.

do_tolower() [3/4]

```
virtual char_type std::ctype< wchar_t >::do_tolower (
            char_type __c) const  [protected], [virtual]
```

Convert to lowercase.
This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.
do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

_↩ _c	The char_type to convert.

Returns

The lowercase char_type if convertible, else __c.

Implements std::__ctype_abstract_base< wchar_t >.

do_tolower() [4/4]

```
virtual char_type std::ctype< wchar_t >::do_tolower (
            char_type __c) const  [protected], [virtual]
```

Convert to lowercase.
This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.
do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

_↩ _c	The wchar_t to convert.

Returns

The lowercase wchar_t if convertible, else __c.

Implements std::__ctype_abstract_base< wchar_t >.

do_toupper() [1/4]

```
virtual const char_type * std::ctype< wchar_t >::do_toupper (
            char_type * __lo,
            const char_type * __hi) const  [protected], [virtual]
```

Convert array to uppercase.
This virtual function converts each char_type in the range [__lo,__hi) to uppercase if possible. Other elements remain untouched.
do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

Implements std::__ctype_abstract_base< wchar_t >.

do_toupper() [2/4]

```
virtual const char_type * std::ctype< wchar_t >::do_toupper (
            char_type * __lo,
            const char_type * __hi) const  [protected], [virtual]
```

Convert array to uppercase.
This virtual function converts each wchar_t in the range [lo,hi) to uppercase if possible. Other elements remain untouched.
do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

Implements std::__ctype_abstract_base< wchar_t >.

do_toupper() [3/4]

```
virtual char_type std::ctype< wchar_t >::do_toupper (
            char_type __c) const  [protected], [virtual]
```

Convert to uppercase.
This virtual function converts the char_type argument to uppercase if possible. If not possible (for example, '2'), returns the argument.
do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

_↩ _c	The char_type to convert.

Returns

The uppercase char_type if convertible, else *__c*.

Implements std::__ctype_abstract_base< wchar_t >.

do_toupper() [4/4]

```
virtual char_type std::ctype< wchar_t >::do_toupper (
            char_type __c) const  [protected], [virtual]
```

Convert to uppercase.
This virtual function converts the wchar_t argument to uppercase if possible. If not possible (for example, '2'), returns the argument.
do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

_↩ _c	The wchar_t to convert.

Returns

The uppercase wchar_t if convertible, else __c.

Implements std::__ctype_abstract_base< wchar_t >.

do_widen() [1/4]

```
virtual char_type std::ctype< wchar_t >::do_widen (
            char __c) const  [protected], [virtual]
```

Widen char.
This virtual function converts the char to char_type using the simplest reasonable transformation.
do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _c	The char to convert.

Returns

The converted char_type

Implements std::__ctype_abstract_base< wchar_t >.

do_widen() [2/4]

```
virtual char_type std::ctype< wchar_t >::do_widen (
            char __c) const  [protected], [virtual]
```

Widen char to wchar_t.
This virtual function converts the char to wchar_t using the simplest reasonable transformation. For an underived ctype<wchar_t> facet, the argument will be cast to wchar_t.
do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _c	The char to convert.

Returns

The converted wchar_t.

Implements std::__ctype_abstract_base< wchar_t >.

do_widen() [3/4]

```
virtual const char * std::ctype< wchar_t >::do_widen (
            const char * __lo,
            const char * __hi,
            char_type * __dest) const  [protected], [virtual]
```

Widen char array.
This function converts each char in the input to char_type using the simplest reasonable transformation.
do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _lo	Pointer to start range.
_↩ _hi	Pointer to end of range.
_↩ _to	Pointer to the destination array.

Returns

__hi.

Implements std::__ctype_abstract_base< wchar_t >.

do_widen() [4/4]

```
virtual const char * std::ctype< wchar_t >::do_widen (
            const char * __lo,
            const char * __hi,
            char_type * __to) const  [protected], [virtual]
```

Widen char array to wchar_t array.
This function converts each char in the input to wchar_t using the simplest reasonable transformation. For an underived ctype<wchar_t> facet, the argument will be copied, casting each element to wchar_t.
do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _lo	Pointer to start range.
_↩ _hi	Pointer to end of range.
_↩ _to	Pointer to the destination array.

Returns

__hi.

Implements std::__ctype_abstract_base< wchar_t >.

is() [1/2]

```
const char_type * std::__ctype_abstract_base< wchar_t >::is (
            const char_type * __lo,
            const char_type * __hi,
            mask * __vec) const  [inline]
```

Return a mask array.
This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char_type>::do_is().

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__vec	Pointer to an array of mask storage.

Returns

__hi.

is() [2/2]

```
bool std::__ctype_abstract_base< wchar_t >::is (
            mask __m,
            char_type __c) const  [inline]
```

Test char_type classification.
This function finds a mask M for *__c* and compares it to mask *__m*. It does so by returning the value of ctype<char_↩ type>::do_is().

Parameters

_↩ _c	The char_type to compare the mask of.
_↩ _m	The mask to compare against.

Returns

(M & __m) != 0.

narrow() [1/2]

```
char std::__ctype_abstract_base< wchar_t >::narrow (
            char_type __c,
            char __dfault) const  [inline]
```

Narrow char_type to char.
This function converts the char_type to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. It does so by returning ctype<char_type>::do_narrow(__c).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char_type to convert.

__dfault	Char to return if conversion fails.

Returns

The converted char.

narrow() [2/2]

```
const char_type * std::__ctype_abstract_base< wchar_t >::narrow (
            const char_type * __lo,
            const char_type * __hi,
            char __dfault,
            char * __to) const  [inline]
```

Narrow array to char array.

This function converts each char_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char_type in the input that cannot be converted, *dfault* is used instead. It does so by returning ctype<char_type>::do_narrow(__lo, __hi, __dfault, __to).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

scan_is()

```
const char_type * std::__ctype_abstract_base< wchar_t >::scan_is (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [inline]
```

Find char_type matching a mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char_type>::do_scan_is().

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to matching char_type if found, else *__hi*.

scan_not()

```
const char_type * std::__ctype_abstract_base< wchar_t >::scan_not (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [inline]
```

Find char_type not matching a mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char_type>::do_scan_not().

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to first char in range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to non-matching char if found, else *__hi*.

tolower() [1/2]

```
const char_type * std::__ctype_abstract_base< wchar_t >::tolower (
            char_type * __lo,
            const char_type * __hi) const  [inline]
```

Convert array to lowercase.

This function converts each char_type in the range [__lo,__hi) to lowercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>:: do_tolower(__lo, __hi).

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

tolower() [2/2]

```
char_type std::__ctype_abstract_base< wchar_t >::tolower (
            char_type __c) const  [inline]
```

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_tolower(c).

Parameters

_↩ _c	The char_type to convert.

Returns

The lowercase char_type if convertible, else __c.

toupper() [1/2]

```
const char_type * std::__ctype_abstract_base< wchar_t >::toupper (
            char_type * __lo,
            const char_type * __hi) const  [inline]
```

Convert array to uppercase.
This function converts each char_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>:: do_toupper(lo, hi).

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

toupper() [2/2]

```
char_type std::__ctype_abstract_base< wchar_t >::toupper (
            char_type __c) const  [inline]
```

Convert to uppercase.
This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_toupper().

Parameters

_↩ _c	The char_type to convert.

Returns

The uppercase char_type if convertible, else __c.

widen() [1/2]

```
char_type std::__ctype_abstract_base< wchar_t >::widen (
            char __c) const  [inline]
```

Widen char to char_type.
This function converts the char argument to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _c	The char to convert.

Returns

The converted char_type.

widen() [2/2]

```
const char * std::__ctype_abstract_base< wchar_t >::widen (
            const char * __lo,
            const char * __hi,
            char_type * __to) const  [inline]
```

Widen array to char_type.
This function converts each char in the input to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.
_↩ _to	Pointer to the destination array.

Returns

__hi.

5.357.5 Member Data Documentation

id [1/2]

```
locale::id std::ctype< wchar_t >::id  [static]
```

The facet id for ctype<char_type>

id [2/2]

```
locale::id std::ctype< wchar_t >::id  [static]
```

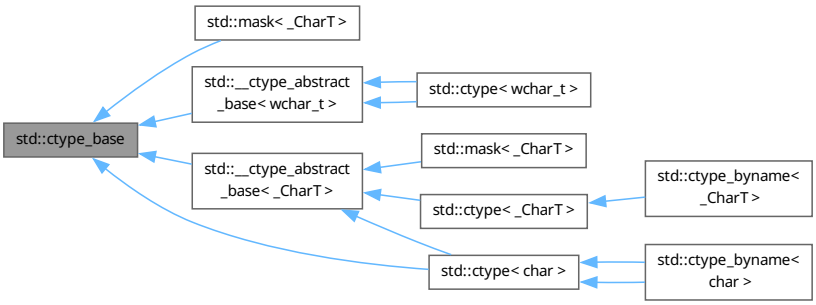
The facet id for ctype<wchar_t>
The documentation for this class was generated from the following file:

- locale_facets.h

5.358 std::ctype_base Struct Reference

```
#include <ctype_base.h>
```

Inheritance diagram for std::ctype_base:



Public Types

- typedef const int * **__to_type**
- typedef unsigned short **mask**

Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **blank**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

5.358.1 Detailed Description

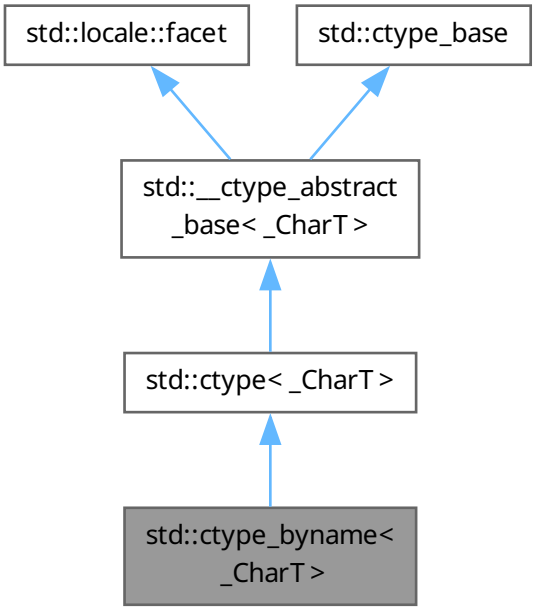
Base class for ctype.
The documentation for this struct was generated from the following file:

- ctype_base.h

5.359 std::ctype_byname< _CharT > Class Template Reference

`#include <locale_facets.h>`

Inheritance diagram for std::ctype_byname< _CharT >:



Public Types

- typedef const int * **__to_type**
- typedef _CharT **char_type**
- typedef ctype< _CharT >::mask **mask**

Public Member Functions

- **ctype_byname** (const char *__s, size_t __refs=0)
- **ctype_byname** (const string &__s, size_t __refs=0)
- const char_type * is (const char_type *__lo, const char_type *__hi, mask *__vec) const
- bool is (mask __m, char_type __c) const
- char narrow (char_type __c, char __dfault) const
- const char_type * narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__to) const
- const char_type * scan_is (mask __m, const char_type *__lo, const char_type *__hi) const
- const char_type * scan_not (mask __m, const char_type *__lo, const char_type *__hi) const
- const char_type * tolower (char_type *__lo, const char_type *__hi) const
- char_type tolower (char_type __c) const
- const char_type * toupper (char_type *__lo, const char_type *__hi) const
- char_type toupper (char_type __c) const
- char_type widen (char __c) const
- const char * widen (const char *__lo, const char *__hi, char_type *__to) const

Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **blank**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static locale::id id
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

Protected Member Functions

- virtual const char_type * do_is (const char_type *__lo, const char_type *__hi, mask *__vec) const
- virtual bool do_is (mask __m, char_type __c) const
- virtual char do_narrow (char_type, char __dfault) const
- virtual const char_type * do_narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__to) const
- virtual const char_type * do_scan_is (mask __m, const char_type *__lo, const char_type *__hi) const
- virtual const char_type * do_scan_not (mask __m, const char_type *__lo, const char_type *__hi) const
- virtual const char_type * do_tolower (char_type *__lo, const char_type *__hi) const
- virtual char_type do_tolower (char_type __c) const
- virtual const char_type * do_toupper (char_type *__lo, const char_type *__hi) const
- virtual char_type do_toupper (char_type __c) const
- virtual char_type do_widen (char __c) const
- virtual const char * do_widen (const char *__lo, const char *__hi, char_type *__dest) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

5.359.1 Detailed Description

template<typename _CharT>
class std::ctype_byname< _CharT >

class ctype_byname [22.2.1.2].

5.359.2 Member Function Documentation

do_is() [1/2]

```
template<typename _CharT>
virtual const char_type * std::ctype< _CharT >::do_is (
            const char_type * __lo,
            const char_type * __hi,
            mask * __vec) const  [protected], [virtual], [inherited]
```

Return a mask array.
This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.
do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__vec	Pointer to an array of mask storage.

Returns

__hi.

Implements std::__ctype_abstract_base< _CharT >.

do_is() [2/2]

```
template<typename _CharT>
virtual bool std::ctype< _CharT >::do_is (
            mask __m,
            char_type __c) const  [protected], [virtual], [inherited]
```

Test char_type classification.
This function finds a mask M for *c* and compares it to mask *m*.
do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

_↩ _c	The char_type to find the mask of.
_↩ _m	The mask to compare against.

Returns

(M & __m) != 0.

Implements std::__ctype_abstract_base< _CharT >.

do_narrow() [1/2]

```
template<typename _CharT>
virtual char std::ctype< _CharT >::do_narrow (
```

```
            char_type __c,
            char __dfault) const  [protected], [virtual], [inherited]
```

Narrow char_type to char.
This virtual function converts the argument to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead.
do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char_type to convert.
__dfault	Char to return if conversion fails.

Returns

The converted char.

Implements std::__ctype_abstract_base< _CharT >.
Referenced by std::ctype< char >::narrow(), and std::ctype< char >::narrow().

do_narrow() [2/2]

```
template<typename _CharT>
virtual const char_type * std::ctype< _CharT >::do_narrow (
            const char_type * __lo,
            const char_type * __hi,
            char __dfault,
            char * __to) const  [protected], [virtual], [inherited]
```

Narrow char_type array to char.
This virtual function converts each char_type in the range [__lo,__hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, *__dfault* is used instead.
do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

Implements std::__ctype_abstract_base< _CharT >.

do_scan_is()

```
template<typename _CharT>
virtual const char_type * std::ctype< _CharT >::do_scan_is (
```

```
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [protected], [virtual], [inherited]
```

Find char_type matching mask.
This function searches for and returns the first char_type c in [__lo,__hi) for which is(__m,c) is true.
do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

_↩ _*m*	The mask to compare against.
_↩ _*lo*	Pointer to start of range.
_↩ _*hi*	Pointer to end of range.

Returns

Pointer to a matching char_type if found, else __*hi*.

Implements std::__ctype_abstract_base< _CharT >.

do_scan_not()

```
template<typename _CharT>
virtual const char_type * std::ctype< _CharT >::do_scan_not (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [protected], [virtual], [inherited]
```

Find char_type not matching mask.
This function searches for and returns a pointer to the first char_type c of [lo,hi) for which is(m,c) is false.
do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

_↩ _*m*	The mask to compare against.
_↩ _*lo*	Pointer to start of range.
_↩ _*hi*	Pointer to end of range.

Returns

Pointer to a non-matching char_type if found, else __*hi*.

Implements std::__ctype_abstract_base< _CharT >.

do_tolower() [1/2]

```
template<typename _CharT>
virtual const char_type * std::ctype< _CharT >::do_tolower (
```

```
            char_type * __lo,
            const char_type * __hi) const  [protected], [virtual], [inherited]
```

Convert array to lowercase.

This virtual function converts each char_type in the range [__lo,__hi) to lowercase if possible. Other elements remain untouched.

do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

__hi.

Implements std::__ctype_abstract_base< _CharT >.

do_tolower() [2/2]

```
template<typename _CharT>
virtual char_type std::ctype< _CharT >::do_tolower (
            char_type __c) const  [protected], [virtual], [inherited]
```

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

__c	The char_type to convert.

Returns

The lowercase char_type if convertible, else *__c*.

Implements std::__ctype_abstract_base< _CharT >.

Referenced by std::ctype< char >::tolower(), and std::ctype< char >::tolower().

do_toupper() [1/2]

```
template<typename _CharT>
virtual const char_type * std::ctype< _CharT >::do_toupper (
            char_type * __lo,
            const char_type * __hi) const  [protected], [virtual], [inherited]
```

Convert array to uppercase.

This virtual function converts each char_type in the range [__lo,__hi) to uppercase if possible. Other elements remain untouched.

do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

Implements std::__ctype_abstract_base< _CharT >.

do_toupper() [2/2]

```
template<typename _CharT>
virtual char_type std::ctype< _CharT >::do_toupper (
            char_type __c) const  [protected], [virtual], [inherited]
```

Convert to uppercase.
This virtual function converts the char_type argument to uppercase if possible. If not possible (for example, '2'), returns the argument.
do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

_↩ _c	The char_type to convert.

Returns

The uppercase char_type if convertible, else *__c*.

Implements std::__ctype_abstract_base< _CharT >.
Referenced by std::ctype< char >::toupper(), and std::ctype< char >::toupper().

do_widen() [1/2]

```
template<typename _CharT>
virtual char_type std::ctype< _CharT >::do_widen (
            char __c) const  [protected], [virtual], [inherited]
```

Widen char.
This virtual function converts the char to char_type using the simplest reasonable transformation.
do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _c	The char to convert.

Returns

The converted char_type

Implements std::__ctype_abstract_base< _CharT >.
Referenced by std::ctype< char >::widen(), and std::ctype< char >::widen().

do_widen() [2/2]

```
template<typename _CharT>
virtual const char * std::ctype< _CharT >::do_widen (
            const char * __lo,
            const char * __hi,
            char_type * __to) const  [protected], [virtual], [inherited]
```

Widen char array.
This function converts each char in the input to char_type using the simplest reasonable transformation.
do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _lo	Pointer to start range.
_↩ _hi	Pointer to end of range.
_↩ _to	Pointer to the destination array.

Returns

__hi.

Implements std::__ctype_abstract_base< _CharT >.

is() [1/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::is (
            const char_type * __lo,
            const char_type * __hi,
            mask * __vec) const  [inline], [inherited]
```

Return a mask array.
This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char_type>::do_is().

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__vec	Pointer to an array of mask storage.

Returns

__hi.

is() [2/2]

```
template<typename _CharT>
bool std::__ctype_abstract_base< _CharT >::is (
```

```
            mask __m,
            char_type __c) const  [inline], [inherited]
```

Test char_type classification.
This function finds a mask M for __c and compares it to mask __m. It does so by returning the value of ctype<char_↩ type>::do_is().

Parameters

_↩ _c	The char_type to compare the mask of.
_↩ _m	The mask to compare against.

Returns

(M & __m) != 0.

Referenced by std::time_get< _CharT, _InIter >::get().

narrow() [1/2]

```
template<typename _CharT>
char std::__ctype_abstract_base< _CharT >::narrow (
            char_type __c,
            char __dfault) const  [inline], [inherited]
```

Narrow char_type to char.
This function converts the char_type to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. It does so by returning ctype<char_type>::do_narrow(__c).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char_type to convert.
__dfault	Char to return if conversion fails.

Returns

The converted char.

Referenced by std::time_get< _CharT, _InIter >::do_get_year(), std::time_get< _CharT, _InIter >::get(), and std::time_put< _CharT, _OutIter >::put().

narrow() [2/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::narrow (
            const char_type * __lo,
            const char_type * __hi,
            char __dfault,
            char * __to) const  [inline], [inherited]
```

Narrow array to char array.
This function converts each char_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char_type in the input that cannot be converted, *dfault* is used instead. It does so by returning ctype<char_type>::do_narrow(__lo, __hi, __dfault, __to).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

scan_is()

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::scan_is (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [inline], [inherited]
```

Find char_type matching a mask.
This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char_type>::do_scan_is().

Parameters

__m	The mask to compare against.
__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

Pointer to matching char_type if found, else *__hi*.

scan_not()

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::scan_not (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [inline], [inherited]
```

Find char_type not matching a mask.
This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char_type>::do_scan_not().

Parameters

__m	The mask to compare against.
__lo	Pointer to first char in range.

_↩ _hi	Pointer to end of range.

Returns

Pointer to non-matching char if found, else __hi.

tolower() [1/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::tolower (
            char_type * __lo,
            const char_type * __hi) const  [inline], [inherited]
```

Convert array to lowercase.

This function converts each char_type in the range [__lo,__hi) to lowercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>:: do_tolower(__lo, __hi).

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

tolower() [2/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::tolower (
            char_type __c) const  [inline], [inherited]
```

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_tolower(c).

Parameters

_↩ _c	The char_type to convert.

Returns

The lowercase char_type if convertible, else __c.

Referenced by std::time_get< _CharT, _InIter >::get().

toupper() [1/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::toupper (
            char_type * __lo,
            const char_type * __hi) const  [inline], [inherited]
```

Convert array to uppercase.
This function converts each char_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>:: do_toupper(lo, hi).

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

toupper() [2/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::toupper (
            char_type __c) const  [inline], [inherited]
```

Convert to uppercase.
This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_toupper().

Parameters

_↩ _c	The char_type to convert.

Returns

The uppercase char_type if convertible, else *__c*.

Referenced by std::time_get< _CharT, _InIter >::get().

widen() [1/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::widen (
            char __c) const  [inline], [inherited]
```

Widen char to char_type.
This function converts the char argument to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _c	The char to convert.

Returns

The converted char_type.

Referenced by std::money_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::money_put< _CharT, _Out and std::time_put< _CharT, _OutIter >::do_put().

widen() [2/2]

```
template<typename _CharT>
const char * std::__ctype_abstract_base< _CharT >::widen (
             const char * __lo,
             const char * __hi,
             char_type * __to) const  [inline], [inherited]
```

Widen array to char_type.
This function converts each char in the input to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.
_↩ _to	Pointer to the destination array.

Returns

__hi.

5.359.3 Member Data Documentation

id

```
template<typename _CharT>
locale::id std::ctype< _CharT >::id  [static], [inherited]
```

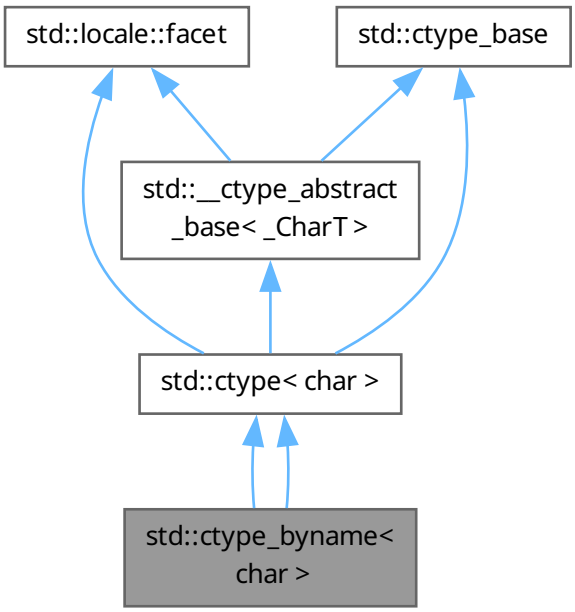
The facet id for ctype<char_type>
The documentation for this class was generated from the following file:

- locale_facets.h

5.360 std::ctype_byname< char > Class Reference

```
#include <locale_facets.h>
```

Inheritance diagram for std::ctype_byname< char >:



Public Types

- typedef const int * **__to_type**
- typedef char **char_type**
- typedef ctype< char >::mask **mask**

Public Member Functions

- **ctype_byname** (const char *__s, size_t __refs=0)
- **ctype_byname** (const char *__s, size_t __refs=0)
- **ctype_byname** (const string &__s, size_t __refs=0)
- **ctype_byname** (const string &__s, size_t __refs=0)
- const char_type * is (const char_type *__lo, const char_type *__hi, mask *__vec) const
- bool is (mask __m, char_type __c) const
- const char_type * is (const char_type *__lo, const char_type *__hi, mask *__vec) const
- bool is (mask __m, char_type __c) const
- char narrow (char_type __c, char __dfault) const
- const char_type * narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__to) const
- char narrow (char_type __c, char __dfault) const
- const char_type * narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__to) const
- const char_type * scan_is (mask __m, const char_type *__lo, const char_type *__hi) const
- const char_type * scan_is (mask __m, const char_type *__lo, const char_type *__hi) const
- const char_type * scan_not (mask __m, const char_type *__lo, const char_type *__hi) const
- const char_type * scan_not (mask __m, const char_type *__lo, const char_type *__hi) const

- const mask * table () const throw ()
- const char_type * tolower (char_type *__lo, const char_type *__hi) const
- char_type tolower (char_type __c) const
- const char_type * tolower (char_type *__lo, const char_type *__hi) const
- char_type tolower (char_type __c) const
- const char_type * toupper (char_type *__lo, const char_type *__hi) const
- char_type toupper (char_type __c) const
- const char_type * toupper (char_type *__lo, const char_type *__hi) const
- char_type toupper (char_type __c) const
- const char * widen (const char *__lo, const char *__hi, char_type *__to) const
- char_type widen (char __c) const
- const char * widen (const char *__lo, const char *__hi, char_type *__to) const

Static Public Member Functions

- static const mask * classic_table () throw ()

Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **blank**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static locale::id id
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const size_t table_size
- static const mask **upper**
- static const mask **xdigit**

Protected Member Functions

- virtual const char_type * do_is (const char_type *__lo, const char_type *__hi, mask *__vec) const
- virtual bool do_is (mask __m, char_type __c) const
- virtual const char_type * do_is (const char_type *__lo, const char_type *__hi, mask *__vec) const
- virtual bool do_is (mask __m, char_type __c) const
- virtual char do_narrow (char_type __c, char __dfault) const
- virtual const char_type * do_narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__to) const
- virtual char do_narrow (char_type, char __dfault) const
- virtual const char_type * do_narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__to) const
- virtual const char_type * do_scan_is (mask __m, const char_type *__lo, const char_type *__hi) const
- virtual const char_type * do_scan_is (mask __m, const char_type *__lo, const char_type *__hi) const
- virtual const char_type * do_scan_not (mask __m, const char_type *__lo, const char_type *__hi) const
- virtual const char_type * do_scan_not (mask __m, const char_type *__lo, const char_type *__hi) const
- virtual const char_type * do_tolower (char_type *__lo, const char_type *__hi) const
- virtual char_type do_tolower (char_type __c) const

- virtual const char_type * do_tolower (char_type *__lo, const char_type *__hi) const
- virtual char_type do_tolower (char_type __c) const
- virtual const char_type * do_toupper (char_type *__lo, const char_type *__hi) const
- virtual char_type do_toupper (char_type __c) const
- virtual const char_type * do_toupper (char_type *__lo, const char_type *__hi) const
- virtual char_type do_toupper (char_type __c) const
- virtual const char * do_widen (const char *__lo, const char *__hi, char_type *__to) const
- virtual char_type do_widen (char __c) const
- virtual const char * do_widen (const char *__lo, const char *__hi, char_type *__dest) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale **_M_c_locale_ctype**
- bool **_M_del**
- char **_M_narrow** [1+static_cast< unsigned char >(-1)]
- char **_M_narrow_ok**
- const mask * **_M_table**
- __to_type **_M_tolower**
- __to_type **_M_toupper**
- char **_M_widen** [1+static_cast< unsigned char >(-1)]
- char **_M_widen_ok**

5.360.1 Detailed Description

22.2.1.4 Class ctype_byname specializations.

5.360.2 Member Function Documentation

classic_table()

```
static const mask * std::ctype< char >::classic_table () throw ( )   [static], [inherited]
```

Returns a pointer to the C locale mask table.

do_is() [1/4]

```
virtual const char_type * std::ctype< char >::do_is (
            const char_type * __lo,
            const char_type * __hi,
            mask * __vec) const  [protected], [virtual], [inherited]
```

Return a mask array.

This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.

do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__vec	Pointer to an array of mask storage.

Returns

__hi.

do_is() [2/4]

```
virtual bool std::ctype< char >::do_is (
            mask __m,
            char_type __c) const  [protected], [virtual], [inherited]
```

Test char_type classification.
This function finds a mask M for *c* and compares it to mask *m*.
do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

__c	The char_type to find the mask of.
__m	The mask to compare against.

Returns

(M & __m) != 0.

do_is() [3/4]

```
virtual const char_type * std::ctype< char >::do_is (
            const char_type * __lo,
            const char_type * __hi,
            mask * __vec) const  [protected], [virtual]
```

Return a mask array.
This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.
do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__vec	Pointer to an array of mask storage.

Returns

__hi.

do_is() [4/4]

```
virtual bool std::ctype< char >::do_is (
            mask __m,
            char_type __c) const  [protected], [virtual]
```

Test char_type classification.
This function finds a mask M for *c* and compares it to mask *m*.
do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

__c	The char_type to find the mask of.
__m	The mask to compare against.

Returns

(M & __m) != 0.

do_narrow() [1/4]

```
virtual char std::ctype< char >::do_narrow (
            char_type __c,
            char __dfault) const  [inline], [protected], [virtual], [inherited]
```

Narrow char.
This virtual function converts the char to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. For an underived ctype<char> facet, *c* will be returned unchanged.
do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char to convert.
__dfault	Char to return if conversion fails.

Returns

The converted char.

do_narrow() [2/4]

```
virtual const char_type * std::ctype< char >::do_narrow (
            const char_type * __lo,
            const char_type * __hi,
            char __dfault,
            char * __to) const  [inline], [protected], [virtual], [inherited]
```

Narrow char array to char array.
This virtual function converts each char in the range [lo,hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *dfault* is used instead. For an underived ctype<char> facet, the argument will be copied unchanged.

do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

do_narrow() [3/4]

```
virtual char std::ctype< char >::do_narrow (
            char_type ,
            char __dfault) const  [protected], [virtual]
```

Narrow char_type to char.
This virtual function converts the argument to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead.
do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char_type to convert.
__dfault	Char to return if conversion fails.

Returns

The converted char.

do_narrow() [4/4]

```
virtual const char_type * std::ctype< char >::do_narrow (
            const char_type * __lo,
            const char_type * __hi,
            char __dfault,
            char * __to) const  [protected], [virtual]
```

Narrow char_type array to char.
This virtual function converts each char_type in the range [__lo,__hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, *__dfault* is used instead.
do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

do_scan_is() [1/2]

```
virtual const char_type * std::ctype< char >::do_scan_is (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [protected], [virtual], [inherited]
```

Find char_type matching mask.
This function searches for and returns the first char_type c in [__lo,__hi) for which is(__m,c) is true.
do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

__m	The mask to compare against.
__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

Pointer to a matching char_type if found, else *__hi*.

do_scan_is() [2/2]

```
virtual const char_type * std::ctype< char >::do_scan_is (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [protected], [virtual]
```

Find char_type matching mask.
This function searches for and returns the first char_type c in [__lo,__hi) for which is(__m,c) is true.
do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

__m	The mask to compare against.
__lo	Pointer to start of range.

_↩ _*hi*	Pointer to end of range.

Returns

Pointer to a matching char_type if found, else __*hi*.

do_scan_not() [1/2]

```
virtual const char_type * std::ctype< char >::do_scan_not (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [protected], [virtual], [inherited]
```

Find char_type not matching mask.
This function searches for and returns a pointer to the first char_type c of [lo,hi) for which is(m,c) is false.
do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

_↩ _*m*	The mask to compare against.
_↩ _*lo*	Pointer to start of range.
_↩ _*hi*	Pointer to end of range.

Returns

Pointer to a non-matching char_type if found, else __*hi*.

do_scan_not() [2/2]

```
virtual const char_type * std::ctype< char >::do_scan_not (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [protected], [virtual]
```

Find char_type not matching mask.
This function searches for and returns a pointer to the first char_type c of [lo,hi) for which is(m,c) is false.
do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

_↩ _*m*	The mask to compare against.
_↩ _*lo*	Pointer to start of range.
_↩ _*hi*	Pointer to end of range.

Returns

Pointer to a non-matching char_type if found, else *__hi*.

do_tolower() [1/4]

```
virtual const char_type * std::ctype< char >::do_tolower (
            char_type * __lo,
            const char_type * __hi) const  [protected], [virtual], [inherited]
```

Convert array to lowercase.
This virtual function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.
do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

__lo	Pointer to first char in range.
__hi	Pointer to end of range.

Returns

__hi.

References do_tolower().

do_tolower() [2/4]

```
virtual char_type std::ctype< char >::do_tolower (
            char_type __c) const  [protected], [virtual], [inherited]
```

Convert to lowercase.
This virtual function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.
do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

__c	The char to convert.

Returns

The lowercase char if convertible, else *__c*.

References do_tolower().
Referenced by do_tolower(), and do_tolower().

do_tolower() [3/4]

```
virtual const char_type * std::ctype< char >::do_tolower (
            char_type * __lo,
            const char_type * __hi) const  [protected], [virtual]
```

Convert array to lowercase.

This virtual function converts each char_type in the range [__lo,__hi) to lowercase if possible. Other elements remain untouched.
do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

__hi.

do_tolower() [4/4]

```
virtual char_type std::ctype< char >::do_tolower (
            char_type __c) const  [protected], [virtual]
```

Convert to lowercase.
This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.
do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

__c	The char_type to convert.

Returns

The lowercase char_type if convertible, else *__c*.

do_toupper() [1/4]

```
virtual const char_type * std::ctype< char >::do_toupper (
            char_type * __lo,
            const char_type * __hi) const  [protected], [virtual], [inherited]
```

Convert array to uppercase.
This virtual function converts each char in the range [lo,hi) to uppercase if possible. Other chars remain untouched.
do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

__hi.

References do_toupper().

do_toupper() [2/4]

```
virtual char_type std::ctype< char >::do_toupper (
            char_type __c) const  [protected], [virtual], [inherited]
```

Convert to uppercase.
This virtual function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.
do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

__c	The char to convert.

Returns

The uppercase char if convertible, else *__c*.

References do_toupper().
Referenced by do_toupper(), and do_toupper().

do_toupper() [3/4]

```
virtual const char_type * std::ctype< char >::do_toupper (
            char_type * __lo,
            const char_type * __hi) const  [protected], [virtual]
```

Convert array to uppercase.
This virtual function converts each char_type in the range [__lo,__hi) to uppercase if possible. Other elements remain untouched.
do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

__hi.

do_toupper() [4/4]

```
virtual char_type std::ctype< char >::do_toupper (
            char_type __c) const  [protected], [virtual]
```

Convert to uppercase.

This virtual function converts the char_type argument to uppercase if possible. If not possible (for example, '2'), returns the argument.
do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

_↩ _c	The char_type to convert.

Returns

The uppercase char_type if convertible, else __c.

do_widen() [1/3]

```
virtual const char * std::ctype< char >::do_widen (
            const char * __lo,
            const char * __hi,
            char_type * __to) const  [inline], [protected], [virtual], [inherited]
```

Widen char array.
This function converts each char in the range [lo,hi) to char using the simplest reasonable transformation. For an underived ctype<char> facet, the argument will be copied unchanged.
do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.
_↩ _to	Pointer to the destination array.

Returns

__hi.

do_widen() [2/3]

```
virtual char_type std::ctype< char >::do_widen (
            char __c) const  [protected], [virtual]
```

Widen char.
This virtual function converts the char to char_type using the simplest reasonable transformation.
do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _c	The char to convert.

Returns

The converted char_type

do_widen() [3/3]

```
virtual const char * std::ctype< char >::do_widen (
            const char * __lo,
            const char * __hi,
            char_type * __dest) const  [protected], [virtual]
```

Widen char array.
This function converts each char in the input to char_type using the simplest reasonable transformation.
do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _lo	Pointer to start range.
_↩ _hi	Pointer to end of range.
_↩ _to	Pointer to the destination array.

Returns

__hi.

is() [1/4]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::is (
            const char_type * __lo,
            const char_type * __hi,
            mask * __vec) const  [inline], [inherited]
```

Return a mask array.
This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char_type>::do_is().

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__vec	Pointer to an array of mask storage.

Returns

__hi.

is() [2/4]

```
template<typename _CharT>
bool std::__ctype_abstract_base< _CharT >::is (
            mask __m,
            char_type __c) const  [inline], [inherited]
```

Test char_type classification.
This function finds a mask M for __c and compares it to mask __m. It does so by returning the value of ctype<char_↩ type>::do_is().

Parameters

_↩ _c	The char_type to compare the mask of.
_↩ _m	The mask to compare against.

Returns

(M & __m) != 0.

Referenced by std::time_get< _CharT, _InIter >::get().

is() [3/4]

```
const char_type * std::__ctype_abstract_base< char >::is (
            const char_type * __lo,
            const char_type * __hi,
            mask * __vec) const  [inline]
```

Return a mask array.
This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char_type>::do_is().

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__vec	Pointer to an array of mask storage.

Returns

__hi.

is() [4/4]

```
bool std::__ctype_abstract_base< char >::is (
            mask __m,
            char_type __c) const  [inline]
```

Test char_type classification.
This function finds a mask M for __c and compares it to mask __m. It does so by returning the value of ctype<char_↩ type>::do_is().

Parameters

__c	The char_type to compare the mask of.
__m	The mask to compare against.

Returns

(M & __m) != 0.

narrow() [1/4]

```
template<typename _CharT>
char std::__ctype_abstract_base< _CharT >::narrow (
            char_type __c,
            char __dfault) const  [inline], [inherited]
```

Narrow char_type to char.
This function converts the char_type to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. It does so by returning ctype<char_type>::do_narrow(__c).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char_type to convert.
__dfault	Char to return if conversion fails.

Returns

The converted char.

Referenced by std::time_get< _CharT, _InIter >::do_get_year(), std::time_get< _CharT, _InIter >::get(), and std::time_put< _CharT, _OutIter >::put().

narrow() [2/4]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::narrow (
            const char_type * __lo,
            const char_type * __hi,
            char __dfault,
            char * __to) const  [inline], [inherited]
```

Narrow array to char array.
This function converts each char_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char_type in the input that cannot be converted, *dfault* is used instead. It does so by returning ctype<char_type>::do_narrow(__lo, __hi, __dfault, __to).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

narrow() [3/4]

```
char std::__ctype_abstract_base< char >::narrow (
            char_type __c,
            char __dfault) const  [inline]
```

Narrow char_type to char.
This function converts the char_type to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. It does so by returning ctype<char_type>::do_narrow(__c).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char_type to convert.
__dfault	Char to return if conversion fails.

Returns

The converted char.

narrow() [4/4]

```
const char_type * std::__ctype_abstract_base< char >::narrow (
            const char_type * __lo,
            const char_type * __hi,
            char __dfault,
            char * __to) const  [inline]
```

Narrow array to char array.
This function converts each char_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char_type in the input that cannot be converted, *dfault* is used instead. It does so by returning ctype<char_type>::do_narrow(__lo, __hi, __dfault, __to).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

scan_is() [1/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::scan_is (
            mask __m,
```

```
            const char_type * __lo,
            const char_type * __hi) const  [inline], [inherited]
```

Find char_type matching a mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char_type>::do_scan_is().

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to matching char_type if found, else *__hi*.

scan_is() [2/2]

```
const char_type * std::__ctype_abstract_base< char >::scan_is (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [inline]
```

Find char_type matching a mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char_type>::do_scan_is().

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to matching char_type if found, else *__hi*.

scan_not() [1/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::scan_not (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [inline], [inherited]
```

Find char_type not matching a mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char_type>::do_scan_not().

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to first char in range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to non-matching char if found, else __hi.

scan_not() [2/2]

```
const char_type * std::__ctype_abstract_base< char >::scan_not (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [inline]
```

Find char_type not matching a mask.
This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char_type>::do_scan_not().

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to first char in range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to non-matching char if found, else __hi.

table()

```
const mask * std::ctype< char >::table () const throw ( )   [inline], [inherited]
```

Returns a pointer to the mask table provided to the constructor, or the default from classic_table() if none was provided.

tolower() [1/4]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::tolower (
            char_type * __lo,
            const char_type * __hi) const  [inline], [inherited]
```

Convert array to lowercase.
This function converts each char_type in the range [__lo,__hi) to lowercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>:: do_tolower(__lo, __hi).

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

tolower() [2/4]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::tolower (
            char_type __c) const  [inline], [inherited]
```

Convert to lowercase.
This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_tolower(c).

Parameters

_↩ _c	The char_type to convert.

Returns

The lowercase char_type if convertible, else __c.

Referenced by std::time_get< _CharT, _InIter >::get().

tolower() [3/4]

```
const char_type * std::__ctype_abstract_base< char >::tolower (
            char_type * __lo,
            const char_type * __hi) const  [inline]
```

Convert array to lowercase.
This function converts each char_type in the range [__lo,__hi) to lowercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>:: do_tolower(__lo, __hi).

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

tolower() [4/4]

```
char_type std::__ctype_abstract_base< char >::tolower (
            char_type __c) const  [inline]
```

Convert to lowercase.
This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_tolower(c).

Parameters

_↩ _c	The char_type to convert.

Returns

The lowercase char_type if convertible, else __c.

toupper() [1/4]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::toupper (
            char_type * __lo,
            const char_type * __hi) const  [inline], [inherited]
```

Convert array to uppercase.
This function converts each char_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>:: do_toupper(lo, hi).

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

toupper() [2/4]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::toupper (
            char_type __c) const  [inline], [inherited]
```

Convert to uppercase.
This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_toupper().

Parameters

_↩ _c	The char_type to convert.

Returns

The uppercase char_type if convertible, else __c.

Referenced by std::time_get< _CharT, _InIter >::get().

toupper() [3/4]

```
const char_type * std::__ctype_abstract_base< char >::toupper (
            char_type * __lo,
            const char_type * __hi) const  [inline]
```

Convert array to uppercase.
This function converts each char_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>:: do_toupper(lo, hi).

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

toupper() [4/4]

```
char_type std::__ctype_abstract_base< char >::toupper (
            char_type __c) const  [inline]
```

Convert to uppercase.
This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_toupper().

Parameters

_↩ _c	The char_type to convert.

Returns

The uppercase char_type if convertible, else *__c*.

widen() [1/3]

```
template<typename _CharT>
const char * std::__ctype_abstract_base< _CharT >::widen (
            const char * __lo,
            const char * __hi,
            char_type * __to) const  [inline], [inherited]
```

Widen array to char_type.
This function converts each char in the input to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _lo	Pointer to start of range.

Parameters

_↩ _hi	Pointer to end of range.
_↩ _to	Pointer to the destination array.

Returns

__hi.

widen() [2/3]

```
char_type std::__ctype_abstract_base< char >::widen (
            char __c) const  [inline]
```

Widen char to char_type.
This function converts the char argument to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _c	The char to convert.

Returns

The converted char_type.

widen() [3/3]

```
const char * std::__ctype_abstract_base< char >::widen (
            const char * __lo,
            const char * __hi,
            char_type * __to) const  [inline]
```

Widen array to char_type.
This function converts each char in the input to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.
_↩ _to	Pointer to the destination array.

Returns

__hi.

5.360.3 Member Data Documentation

id

```
locale::id std::ctype< char >::id  [static]
```
The facet id for ctype<char_type>

table_size

```
const size_t std::ctype< char >::table_size  [static], [inherited]
```
The size of the mask table. It is SCHAR_MAX + 1.
The documentation for this class was generated from the following file:

- locale_facets.h

5.361 __gnu_cxx::debug_allocator< _Alloc > Class Template Reference

```
#include <debug_allocator.h>
```

Public Types

- typedef _Traits::const_pointer **const_pointer**
- typedef _Traits::const_reference **const_reference**
- typedef _Traits::difference_type **difference_type**
- typedef _Traits::pointer **pointer**
- typedef _Traits::reference **reference**
- typedef _Traits::size_type **size_type**
- typedef _Traits::value_type **value_type**

Public Member Functions

- **debug_allocator** (const _Alloc &__a)
- template<typename _Alloc2>
  debug_allocator (const debug_allocator< _Alloc2 > &__a2, typename __convertible< _Alloc2 >::__type=0)
- pointer **allocate** (size_type __n)
- pointer **allocate** (size_type __n, const void *__hint)
- template<typename _Tp, typename... _Args>
  void **construct** (_Tp *__p, _Args &&... __args)
- void **construct** (pointer __p, const value_type &__val)
- void **deallocate** (pointer __p, size_type __n)
- template<typename _Tp>
  void **destroy** (_Tp *__p)
- size_type **max_size** () const throw ()

Friends

- template<typename _Alloc2>
  bool **operator==** (const debug_allocator &__lhs, const debug_allocator< _Alloc2 > &__rhs) noexcept

5.361.1 Detailed Description

template<**typename _Alloc**>
class __gnu_cxx::debug_allocator< **_Alloc** >

A meta-allocator with debugging bits.
This is precisely the allocator defined in the C++03 Standard.
The documentation for this class was generated from the following file:

- debug_allocator.h

5.362 std::decay< _Tp > Struct Template Reference

`#include <type_traits>`

Public Types

- using **type**

5.362.1 Detailed Description

template<**typename _Tp**>
struct std::decay< **_Tp** >

decay
The documentation for this struct was generated from the following file:

- type_traits

5.363 std::decimal::decimal128 Class Reference

`#include <decimal>`

Public Types

- typedef float **__decfloat128**

Public Member Functions

- decimal128 (__decfloat128 __z)
- **decimal128** (decimal32 d32)
- **decimal128** (decimal64 d64)
- **decimal128** (double __r)
- **decimal128** (float __r)
- **decimal128** (int __z)
- **decimal128** (long __z)
- **decimal128** (long double __r)
- **decimal128** (long long __z)
- **decimal128** (unsigned int __z)
- **decimal128** (unsigned long __z)
- **decimal128** (unsigned long long __z)
- __decfloat128 **__getval** (void)
- void **__setval** (__decfloat128 __x)
- **operator long long** () const
- decimal128 & **operator*=** (decimal128 __rhs)
- decimal128 & **operator*=** (decimal32 __rhs)

- decimal128 & **operator*=** (decimal64 __rhs)
- decimal128 & **operator*=** (int __rhs)
- decimal128 & **operator*=** (long __rhs)
- decimal128 & **operator*=** (long long __rhs)
- decimal128 & **operator*=** (unsigned int __rhs)
- decimal128 & **operator*=** (unsigned long __rhs)
- decimal128 & **operator*=** (unsigned long long __rhs)
- decimal128 & **operator++** ()
- decimal128 **operator++** (int)
- decimal128 & **operator+=** (decimal128 __rhs)
- decimal128 & **operator+=** (decimal32 __rhs)
- decimal128 & **operator+=** (decimal64 __rhs)
- decimal128 & **operator+=** (int __rhs)
- decimal128 & **operator+=** (long __rhs)
- decimal128 & **operator+=** (long long __rhs)
- decimal128 & **operator+=** (unsigned int __rhs)
- decimal128 & **operator+=** (unsigned long __rhs)
- decimal128 & **operator+=** (unsigned long long __rhs)
- decimal128 & **operator--** ()
- decimal128 **operator--** (int)
- decimal128 & **operator-=** (decimal128 __rhs)
- decimal128 & **operator-=** (decimal32 __rhs)
- decimal128 & **operator-=** (decimal64 __rhs)
- decimal128 & **operator-=** (int __rhs)
- decimal128 & **operator-=** (long __rhs)
- decimal128 & **operator-=** (long long __rhs)
- decimal128 & **operator-=** (unsigned int __rhs)
- decimal128 & **operator-=** (unsigned long __rhs)
- decimal128 & **operator-=** (unsigned long long __rhs)
- decimal128 & **operator/=** (decimal128 __rhs)
- decimal128 & **operator/=** (decimal32 __rhs)
- decimal128 & **operator/=** (decimal64 __rhs)
- decimal128 & **operator/=** (int __rhs)
- decimal128 & **operator/=** (long __rhs)
- decimal128 & **operator/=** (long long __rhs)
- decimal128 & **operator/=** (unsigned int __rhs)
- decimal128 & **operator/=** (unsigned long __rhs)
- decimal128 & **operator/=** (unsigned long long __rhs)

5.363.1 Detailed Description

3.2.4 Class decimal128.

5.363.2 Constructor & Destructor Documentation

decimal128()

```
std::decimal::decimal128::decimal128 (
            __decfloat128 __z)  [inline]
```

Conforming extension: Conversion from scalar decimal type.
The documentation for this class was generated from the following file:

- decimal

5.364 std::decimal::decimal32 Class Reference

`#include <decimal>`

Public Types

- typedef float **__decfloat32**

Public Member Functions

- decimal32 (__decfloat32 __z)
- **decimal32** (decimal128 __d128)
- **decimal32** (decimal64 __d64)
- **decimal32** (double __r)
- **decimal32** (float __r)
- **decimal32** (int __z)
- **decimal32** (long __z)
- **decimal32** (long double __r)
- **decimal32** (long long __z)
- **decimal32** (unsigned int __z)
- **decimal32** (unsigned long __z)
- **decimal32** (unsigned long long __z)
- __decfloat32 **__getval** (void)
- void **__setval** (__decfloat32 __x)
- **operator long long** () const
- decimal32 & **operator*=** (decimal128 __rhs)
- decimal32 & **operator*=** (decimal32 __rhs)
- decimal32 & **operator*=** (decimal64 __rhs)
- decimal32 & **operator*=** (int __rhs)
- decimal32 & **operator*=** (long __rhs)
- decimal32 & **operator*=** (long long __rhs)
- decimal32 & **operator*=** (unsigned int __rhs)
- decimal32 & **operator*=** (unsigned long __rhs)
- decimal32 & **operator*=** (unsigned long long __rhs)
- decimal32 & **operator++** ()
- decimal32 **operator++** (int)
- decimal32 & **operator+=** (decimal128 __rhs)
- decimal32 & **operator+=** (decimal32 __rhs)
- decimal32 & **operator+=** (decimal64 __rhs)
- decimal32 & **operator+=** (int __rhs)
- decimal32 & **operator+=** (long __rhs)
- decimal32 & **operator+=** (long long __rhs)
- decimal32 & **operator+=** (unsigned int __rhs)
- decimal32 & **operator+=** (unsigned long __rhs)
- decimal32 & **operator+=** (unsigned long long __rhs)
- decimal32 & **operator--** ()
- decimal32 **operator--** (int)
- decimal32 & **operator-=** (decimal128 __rhs)
- decimal32 & **operator-=** (decimal32 __rhs)
- decimal32 & **operator-=** (decimal64 __rhs)
- decimal32 & **operator-=** (int __rhs)
- decimal32 & **operator-=** (long __rhs)

- decimal32 & **operator-=** (long long __rhs)
- decimal32 & **operator-=** (unsigned int __rhs)
- decimal32 & **operator-=** (unsigned long __rhs)
- decimal32 & **operator-=** (unsigned long long __rhs)
- decimal32 & **operator/=** (decimal128 __rhs)
- decimal32 & **operator/=** (decimal32 __rhs)
- decimal32 & **operator/=** (decimal64 __rhs)
- decimal32 & **operator/=** (int __rhs)
- decimal32 & **operator/=** (long __rhs)
- decimal32 & **operator/=** (long long __rhs)
- decimal32 & **operator/=** (unsigned int __rhs)
- decimal32 & **operator/=** (unsigned long __rhs)
- decimal32 & **operator/=** (unsigned long long __rhs)

5.364.1 Detailed Description

3.2.2 Class decimal32.

5.364.2 Constructor & Destructor Documentation

decimal32()

```
std::decimal::decimal32::decimal32 (
            __decfloat32 __z)  [inline]
```

Conforming extension: Conversion from scalar decimal type.
The documentation for this class was generated from the following file:

- decimal

5.365 std::decimal::decimal64 Class Reference

```
#include <decimal>
```

Public Types

- typedef float **__decfloat64**

Public Member Functions

- decimal64 (__decfloat64 __z)
- **decimal64** (decimal128 d128)
- **decimal64** (decimal32 d32)
- **decimal64** (double __r)
- **decimal64** (float __r)
- **decimal64** (int __z)
- **decimal64** (long __z)
- **decimal64** (long double __r)
- **decimal64** (long long __z)
- **decimal64** (unsigned int __z)
- **decimal64** (unsigned long __z)
- **decimal64** (unsigned long long __z)
- __decfloat64 **__getval** (void)
- void **__setval** (__decfloat64 __x)
- **operator long long** () const

- decimal64 & **operator*=** (decimal128 __rhs)
- decimal64 & **operator*=** (decimal32 __rhs)
- decimal64 & **operator*=** (decimal64 __rhs)
- decimal64 & **operator*=** (int __rhs)
- decimal64 & **operator*=** (long __rhs)
- decimal64 & **operator*=** (long long __rhs)
- decimal64 & **operator*=** (unsigned int __rhs)
- decimal64 & **operator*=** (unsigned long __rhs)
- decimal64 & **operator*=** (unsigned long long __rhs)
- decimal64 & **operator++** ()
- decimal64 **operator++** (int)
- decimal64 & **operator+=** (decimal128 __rhs)
- decimal64 & **operator+=** (decimal32 __rhs)
- decimal64 & **operator+=** (decimal64 __rhs)
- decimal64 & **operator+=** (int __rhs)
- decimal64 & **operator+=** (long __rhs)
- decimal64 & **operator+=** (long long __rhs)
- decimal64 & **operator+=** (unsigned int __rhs)
- decimal64 & **operator+=** (unsigned long __rhs)
- decimal64 & **operator+=** (unsigned long long __rhs)
- decimal64 & **operator--** ()
- decimal64 **operator--** (int)
- decimal64 & **operator-=** (decimal128 __rhs)
- decimal64 & **operator-=** (decimal32 __rhs)
- decimal64 & **operator-=** (decimal64 __rhs)
- decimal64 & **operator-=** (int __rhs)
- decimal64 & **operator-=** (long __rhs)
- decimal64 & **operator-=** (long long __rhs)
- decimal64 & **operator-=** (unsigned int __rhs)
- decimal64 & **operator-=** (unsigned long __rhs)
- decimal64 & **operator-=** (unsigned long long __rhs)
- decimal64 & **operator/=** (decimal128 __rhs)
- decimal64 & **operator/=** (decimal32 __rhs)
- decimal64 & **operator/=** (decimal64 __rhs)
- decimal64 & **operator/=** (int __rhs)
- decimal64 & **operator/=** (long __rhs)
- decimal64 & **operator/=** (long long __rhs)
- decimal64 & **operator/=** (unsigned int __rhs)
- decimal64 & **operator/=** (unsigned long __rhs)
- decimal64 & **operator/=** (unsigned long long __rhs)

5.365.1 Detailed Description

3.2.3 Class decimal64.

5.365.2 Constructor & Destructor Documentation

decimal64()

```
std::decimal::decimal64::decimal64 (
            __decfloat64 __z)  [inline]
```

Conforming extension: Conversion from scalar decimal type.
The documentation for this class was generated from the following file:

- decimal

5.366 simd_abi::deduce< _Tp, _Np,... > Struct Template Reference

`#include <simd.h>`

5.366.1 Detailed Description

template<typename _Tp, size_t _Np, typename...>
struct simd_abi::deduce< _Tp, _Np,... >

Template Parameters

_Tp	The requested `value_type` for the elements.
_Np	The requested number of elements.
_Abis	This parameter is ignored, since this implementation cannot make any use of it. Either __a good native ABI is matched and used as `type` alias, or the `fixed_size<_Np>` ABI is used, which internally is built from the best matching native ABIs.

The documentation for this struct was generated from the following file:

- simd.h

5.367 __gnu_pbds::detail::default_comb_hash_fn Struct Reference

`#include <standard_policies.hpp>`

Public Types

- typedef direct_mask_range_hashing type

5.367.1 Detailed Description

Primary template, default_comb_hash_fn.

5.367.2 Member Typedef Documentation

type

`typedef direct_mask_range_hashing __gnu_pbds::detail::default_comb_hash_fn::type`
Dispatched type.
The documentation for this struct was generated from the following file:

- standard_policies.hpp

5.368 std::default_delete< _Tp > Struct Template Reference

`#include <memory>`

Public Member Functions

- constexpr default_delete () noexcept=default
- template<typename _Up, typename = _Require<is_convertible<_Up*, _Tp*>>>
  constexpr default_delete (const default_delete< _Up > &) noexcept
- constexpr void operator() (_Tp *__ptr) const

5.368.1 Detailed Description

template<typename _Tp>
struct std::default_delete< _Tp >

Primary template of default_delete, used by unique_ptr for single objects

Since

C++11

5.368.2 Constructor & Destructor Documentation

default_delete() [1/2]

```
template<typename _Tp>
std::default_delete< _Tp >::default_delete ()  [constexpr], [default], [noexcept]
```

Default constructor.
Referenced by default_delete().

default_delete() [2/2]

```
template<typename _Tp>
template<typename _Up, typename = _Require<is_convertible<_Up*, _Tp*>>>
std::default_delete< _Tp >::default_delete (
            const default_delete< _Up > & )  [inline], [constexpr], [noexcept]
```

Converting constructor.
Allows conversion from a deleter for objects of another type, _Up, only if _Up* is convertible to _Tp*.
References default_delete().

5.368.3 Member Function Documentation

operator()()

```
template<typename _Tp>
void std::default_delete< _Tp >::operator() (
            _Tp * __ptr) const  [inline], [constexpr]
```

Calls `delete __ptr`
The documentation for this struct was generated from the following file:

- unique_ptr.h

5.369 std::default_delete< _Tp[]> Struct Template Reference

```
#include <memory>
```

Public Member Functions

- constexpr default_delete () noexcept=default
- constexpr default_delete () noexcept=default
- constexpr default_delete (const default_delete< _Up > &) noexcept
- template<typename _Up, typename = _Require<is_convertible<_Up(*)[], _Tp(*)[]>>>
  constexpr default_delete (const default_delete< _Up[]> &) noexcept
- constexpr void operator() (_Tp *__ptr) const
- template<typename _Up>
  constexpr enable_if< is_convertible< _Up(*)[], _Tp(*)[]>::value >::type operator() (_Up *__ptr) const

5.369.1 Detailed Description

template<typename _Tp>
struct std::default_delete< _Tp[]>

Specialization of default_delete for arrays, used by `unique_ptr<T[]>`

Since

C++11

5.369.2 Constructor & Destructor Documentation

default_delete() [1/4]

```
template<typename _Tp>
std::default_delete< _Tp[]>::default_delete ()  [constexpr], [default], [noexcept]
```

Default constructor.
Referenced by default_delete().

default_delete() [2/4]

```
template<typename _Tp>
template<typename _Up, typename = _Require<is_convertible<_Up(*)[], _Tp(*)[]>>>
std::default_delete< _Tp[]>::default_delete (
            const default_delete< _Up[]> & )  [inline], [constexpr], [noexcept]
```

Converting constructor.
Allows conversion from a deleter for arrays of another type, such as a const-qualified version of `_Tp`.
Conversions from types derived from `_Tp` are not allowed because it is undefined to `delete[]` an array of derived types through a pointer to the base type.
References default_delete().

default_delete() [3/4]

```
std::default_delete< _Tp >::default_delete ()  [constexpr], [default], [noexcept]
```

Default constructor.

default_delete() [4/4]

```
std::default_delete< _Tp >::default_delete (
            const default_delete< _Up > & )  [inline], [constexpr], [noexcept]
```

Converting constructor.
Allows conversion from a deleter for objects of another type, `_Up`, only if `_Up*` is convertible to `_Tp*`.

5.369.3 Member Function Documentation

operator()() [1/2]

```
void std::default_delete< _Tp >::operator() (
            _Tp * __ptr) const  [inline], [constexpr]
```

Calls `delete __ptr`

operator()() [2/2]

```
template<typename _Tp>
template<typename _Up>
```

```
enable_if< is_convertible< _Up(*)[], _Tp(*)[]>::value >::type std::default_delete< _Tp[]>↩
::operator() (
            _Up * __ptr) const  [inline], [constexpr]
```

Calls `delete[] __ptr`

The documentation for this struct was generated from the following file:

- unique_ptr.h

5.370 __gnu_pbds::detail::default_eq_fn< Key > Struct Template Reference

`#include <standard_policies.hpp>`

Public Types

- typedef std::equal_to< Key > type

5.370.1 Detailed Description

template<typename Key>
struct __gnu_pbds::detail::default_eq_fn< Key >

Primary template, default_eq_fn.

5.370.2 Member Typedef Documentation

type

```
template<typename Key>
typedef std::equal_to<Key> __gnu_pbds::detail::default_eq_fn< Key >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- standard_policies.hpp

5.371 __gnu_pbds::detail::default_hash_fn< Key > Struct Template Reference

`#include <standard_policies.hpp>`

Public Types

- typedef std::tr1::hash< Key > type

5.371.1 Detailed Description

template<typename Key>
struct __gnu_pbds::detail::default_hash_fn< Key >

Primary template, default_hash_fn.

5.371.2 Member Typedef Documentation

type

```
template<typename Key>
typedef std::tr1::hash<Key> __gnu_pbds::detail::default_hash_fn< Key >::type
```

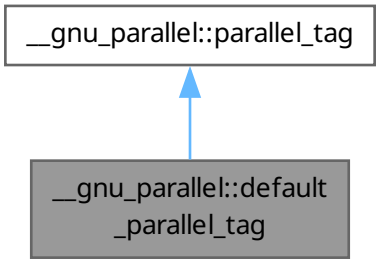
Dispatched type.

The documentation for this struct was generated from the following file:

- standard_policies.hpp

5.372 __gnu_parallel::default_parallel_tag Struct Reference

`#include <tags.h>`
Inheritance diagram for __gnu_parallel::default_parallel_tag:



Public Member Functions

- **default_parallel_tag** (_ThreadIndex __num_threads)
- _ThreadIndex __get_num_threads ()
- void set_num_threads (_ThreadIndex __num_threads)

5.372.1 Detailed Description

Recommends parallel execution using the default parallel algorithm.

5.372.2 Member Function Documentation

__get_num_threads()

`_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads () [inline], [inherited]`
Find out desired number of threads.

Returns

Desired number of threads.

Referenced by __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), and __gnu_parallel::__parallel_sort(

set_num_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
            _ThreadIndex __num_threads) [inline], [inherited]
```
Set the desired number of threads.

Parameters

__num_threads	Desired number of threads.

The documentation for this struct was generated from the following file:

- tags.h

5.373 __gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn > Struct Template Reference

`#include <standard_policies.hpp>`

Public Types

- typedef cond_type::__type type

5.373.1 Detailed Description

template<typename Comb_Probe_Fn>
struct __gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn >

Primary template, default_probe_fn.

5.373.2 Member Typedef Documentation

type

```
template<typename Comb_Probe_Fn>
typedef cond_type::__type __gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn >::type
```

Dispatched type.
The documentation for this struct was generated from the following file:

- standard_policies.hpp

5.374 __gnu_pbds::detail::default_resize_policy< Comb_Hash_Fn > Struct Template Reference

`#include <standard_policies.hpp>`

Public Types

- typedef hash_standard_resize_policy< size_policy_type, trigger, false, size_type > type

5.374.1 Detailed Description

template<typename Comb_Hash_Fn>
struct __gnu_pbds::detail::default_resize_policy< Comb_Hash_Fn >

Primary template, default_resize_policy.

5.374.2 Member Typedef Documentation

type

```
template<typename Comb_Hash_Fn>
typedef hash_standard_resize_policy<size_policy_type, trigger, false, size_type> __gnu_pbds::detail::default_resi
Comb_Hash_Fn >::type
```

Dispatched type.
The documentation for this struct was generated from the following file:

- standard_policies.hpp

5.375 std::default_sentinel_t Struct Reference

`#include <iterator_concepts.h>`

5.375.1 Detailed Description

A sentinel type that can be used to check for the end of a range.
For some iterator types the past-the-end sentinel value is independent of the underlying sequence, and a default sentinel can be used with them. For example, a `std::counted_iterator` keeps a count of how many elements remain, and so checking for the past-the-end value only requires checking if that count has reached zero. A past-the-end `std::istream_iterator` is equal to the default-constructed value, which can be easily checked.
Comparing iterators of these types to `std::default_sentinel` is a convenient way to check if the end has been reached.

Since

C++20

The documentation for this struct was generated from the following file:

- iterator_concepts.h

5.376 __gnu_pbds::detail::default_trie_access_traits< Key > Struct Template Reference

5.376.1 Detailed Description

template<typename Key>
struct __gnu_pbds::detail::default_trie_access_traits< Key >

Primary template, default_trie_access_traits.
The documentation for this struct was generated from the following file:

- standard_policies.hpp

5.377 __gnu_pbds::detail::default_trie_access_traits< std::basic_string< Char, Char_Traits, std::allocator< char > > > Struct Template Reference

`#include <standard_policies.hpp>`

Public Types

- typedef trie_string_access_traits< string_type > type

5.377.1 Detailed Description

template<typename Char, typename Char_Traits>
struct __gnu_pbds::detail::default_trie_access_traits< std::basic_string< Char, Char_Traits, std::allocator< char > > >

Partial specialization, default_trie_access_traits.

5.377.2 Member Typedef Documentation

type

```
template<typename Char, typename Char_Traits>
typedef trie_string_access_traits<string_type> __gnu_pbds::detail::default_trie_access_traits<
std::basic_string< Char, Char_Traits, std::allocator< char > > >::type
```

Dispatched type.
The documentation for this struct was generated from the following file:

- standard_policies.hpp

5.378 __gnu_pbds::detail::default_update_policy Struct Reference

`#include <standard_policies.hpp>`

Public Types

- typedef lu_move_to_front_policy type

5.378.1 Detailed Description

Default update policy.

5.378.2 Member Typedef Documentation

type

`typedef lu_move_to_front_policy __gnu_pbds::detail::default_update_policy::type`
Dispatched type.
The documentation for this struct was generated from the following file:

- standard_policies.hpp

5.379 std::defer_lock_t Struct Reference

`#include <std_mutex.h>`

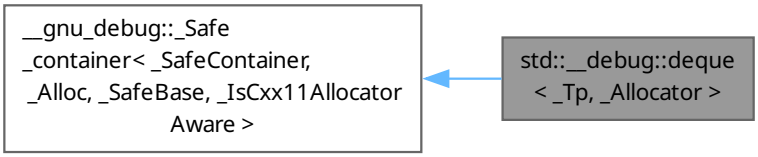
5.379.1 Detailed Description

Do not acquire ownership of the mutex.
The documentation for this struct was generated from the following file:

- std_mutex.h

5.380 std::__debug::deque< _Tp, _Allocator > Class Template Reference

`#include <deque>`
Inheritance diagram for std::__debug::deque< _Tp, _Allocator >:



Public Types

- typedef _Allocator **allocator_type**
- typedef __gnu_debug::_Safe_iterator< _Base_const_iterator, deque > **const_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**

- typedef std::reverse_iterator< const_iterator > **const_reverse_iterator**
- typedef _Base::difference_type **difference_type**
- typedef __gnu_debug::_Safe_iterator< _Base_iterator, deque > **iterator**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef std::reverse_iterator< iterator > **reverse_iterator**
- typedef _Base::size_type **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **deque** (_Base_ref __x)
- template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  deque (_InputIterator __first, _InputIterator __last, const _Allocator &__a=_Allocator())
- **deque** (const _Allocator &__a)
- **deque** (const deque &)=default
- **deque** (const deque &__d, const __type_identity_t< _Allocator > &__a)
- **deque** (deque &&)=default
- **deque** (deque &&__d, const __type_identity_t< _Allocator > &__a)
- **deque** (initializer_list< value_type > __l, const allocator_type &__a=allocator_type())
- **deque** (size_type __n, const __type_identity_t< _Tp > &__value, const _Allocator &__a=_Allocator())
- **deque** (size_type __n, const _Allocator &__a=_Allocator())
- const _Base & **_M_base** () const noexcept
- _Base & **_M_base** () noexcept
- template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  void **assign** (_InputIterator __first, _InputIterator __last)
- void **assign** (initializer_list< value_type > __l)
- void **assign** (size_type __n, const _Tp &__t)
- const_reference **back** () const noexcept
- reference **back** () noexcept
- const_iterator **begin** () const noexcept
- iterator **begin** () noexcept
- const_iterator **cbegin** () const noexcept
- const_iterator **cend** () const noexcept
- void **clear** () noexcept
- const_reverse_iterator **crbegin** () const noexcept
- const_reverse_iterator **crend** () const noexcept
- template<typename... _Args>
  iterator **emplace** (const_iterator __position, _Args &&... __args)
- template<typename... _Args>
  reference **emplace_back** (_Args &&... __args)
- template<typename... _Args>
  reference **emplace_front** (_Args &&... __args)
- const_iterator **end** () const noexcept
- iterator **end** () noexcept
- iterator **erase** (const_iterator __first, const_iterator __last)
- iterator **erase** (const_iterator __position)
- const_reference **front** () const noexcept
- reference **front** () noexcept
- template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  iterator **insert** (const_iterator __position, _InputIterator __first, _InputIterator __last)

- iterator **insert** (const_iterator __position, _Tp &&__x)
- iterator **insert** (const_iterator __position, const _Tp &__x)
- iterator **insert** (const_iterator __position, initializer_list< value_type > __l)
- iterator **insert** (const_iterator __position, size_type __n, const _Tp &__x)
- deque & **operator=** (const deque &)=default
- deque & **operator=** (deque &&)=default
- deque & **operator=** (initializer_list< value_type > __l)
- const_reference **operator[ ]** (size_type __n) const noexcept
- reference **operator[ ]** (size_type __n) noexcept
- void **pop_back** () noexcept
- void **pop_front** () noexcept
- void **push_back** (_Tp &&__x)
- void **push_back** (const _Tp &__x)
- void **push_front** (_Tp &&__x)
- void **push_front** (const _Tp &__x)
- const_reverse_iterator **rbegin** () const noexcept
- reverse_iterator **rbegin** () noexcept
- const_reverse_iterator **rend** () const noexcept
- reverse_iterator **rend** () noexcept
- void **resize** (size_type __sz)
- void **resize** (size_type __sz, const _Tp &__c)
- void **shrink_to_fit** () noexcept
- void **swap** (deque &__x) noexcept(/*conditional */)

Protected Member Functions

- constexpr void **_M_swap** (const _Safe_container &__x) const noexcept

Friends

- template<typename _ItT, typename _SeqT, typename _CatT>
  class **::__gnu_debug::_Safe_iterator**

5.380.1 Detailed Description

template<typename _Tp, typename _Allocator = std::allocator<_Tp>>
class std::__debug::deque< _Tp, _Allocator >

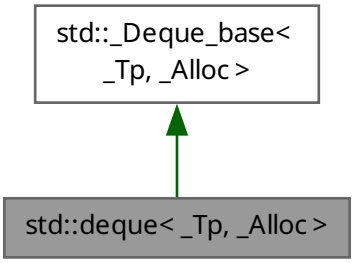
Class std::deque with safety/checking/debug instrumentation.
The documentation for this class was generated from the following file:

- debug/deque

5.381 std::deque< _Tp, _Alloc > Class Template Reference

`#include <stl_deque.h>`

Inheritance diagram for std::deque< _Tp, _Alloc >:



Public Types

- typedef _Alloc **allocator_type**
- typedef _Base::const_iterator **const_iterator**
- typedef _Alloc_traits::const_pointer **const_pointer**
- typedef _Alloc_traits::const_reference **const_reference**
- typedef std::reverse_iterator< const_iterator > **const_reverse_iterator**
- typedef ptrdiff_t **difference_type**
- typedef _Base::iterator **iterator**
- typedef _Alloc_traits::pointer **pointer**
- typedef _Alloc_traits::reference **reference**
- typedef std::reverse_iterator< iterator > **reverse_iterator**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- deque ()=default
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  deque (_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type())
- deque (const allocator_type &__a)
- deque (const deque &__x)
- deque (const deque &__x, const __type_identity_t< allocator_type > &__a)
- deque (deque &&)=default
- deque (deque &&__x, const __type_identity_t< allocator_type > &__a)
- deque (initializer_list< value_type > __l, const allocator_type &__a=allocator_type())
- deque (size_type __n, const allocator_type &__a=allocator_type())
- deque (size_type __n, const value_type &__value, const allocator_type &__a=allocator_type())
- ~deque ()
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  void assign (_InputIterator __first, _InputIterator __last)
- void assign (initializer_list< value_type > __l)
- void assign (size_type __n, const value_type &__val)
- reference at (size_type __n)

- const_reference at (size_type __n) const
- const_reference back () const noexcept
- reference back () noexcept
- const_iterator begin () const noexcept
- iterator begin () noexcept
- const_iterator cbegin () const noexcept
- const_iterator cend () const noexcept
- void clear () noexcept
- const_reverse_iterator crbegin () const noexcept
- const_reverse_iterator crend () const noexcept
- template<typename... _Args>
  iterator emplace (const_iterator __position, _Args &&... __args)
- template<typename... _Args>
  reference **emplace_back** (_Args &&... __args)
- template<typename... _Args>
  reference **emplace_front** (_Args &&... __args)
- bool empty () const noexcept
- const_iterator end () const noexcept
- iterator end () noexcept
- iterator erase (const_iterator __first, const_iterator __last)
- iterator erase (const_iterator __position)
- const_reference front () const noexcept
- reference front () noexcept
- allocator_type get_allocator () const noexcept
- iterator insert (const_iterator __p, initializer_list< value_type > __l)
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  iterator insert (const_iterator __position, _InputIterator __first, _InputIterator __last)
- iterator insert (const_iterator __position, const value_type &__x)
- iterator insert (const_iterator __position, size_type __n, const value_type &__x)
- iterator insert (const_iterator __position, value_type &&__x)
- size_type max_size () const noexcept
- deque & operator= (const deque &__x)
- deque & operator= (deque &&__x) noexcept(_Alloc_traits::_S_always_equal())
- deque & operator= (initializer_list< value_type > __l)
- const_reference operator[ ] (size_type __n) const noexcept
- reference operator[ ] (size_type __n) noexcept
- void pop_back () noexcept
- void pop_front () noexcept
- void push_back (const value_type &__x)
- void **push_back** (value_type &&__x)
- void push_front (const value_type &__x)
- void **push_front** (value_type &&__x)
- const_reverse_iterator rbegin () const noexcept
- reverse_iterator rbegin () noexcept
- const_reverse_iterator rend () const noexcept
- reverse_iterator rend () noexcept
- void resize (size_type __new_size)
- void resize (size_type __new_size, const value_type &__x)
- void shrink_to_fit () noexcept
- size_type size () const noexcept
- void swap (deque &__x) noexcept

Protected Types

- enum { **_S_initial_map_size** }
- typedef __gnu_cxx::__alloc_traits< _Map_alloc_type > **_Map_alloc_traits**
- typedef _Alloc_traits::template rebind< _Ptr >::other **_Map_alloc_type**
- typedef _Alloc_traits::pointer **_Ptr**
- typedef _Alloc_traits::const_pointer **_Ptr_const**

Protected Member Functions

- template<typename _ForwardIterator>
  void **_M_assign_aux** (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- template<typename _InputIterator>
  void **_M_assign_aux** (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- void **_M_deallocate_map** (_Map_pointer __p, size_t __n) noexcept
- void **_M_default_append** (size_type __n)
- void **_M_default_initialize** ()
- template<typename _Alloc1>
  void **_M_destroy_data** (iterator __first, iterator __last, const _Alloc1 &)
- void **_M_destroy_data** (iterator __first, iterator __last, const std::allocator< _Tp > &)
- void **_M_destroy_data_aux** (iterator __first, iterator __last)
- template<typename... _Args>
  iterator **_M_emplace_aux** (iterator __pos, _Args &&... __args)
- iterator **_M_erase** (iterator __first, iterator __last)
- iterator **_M_erase** (iterator __pos)
- void **_M_erase_at_begin** (iterator __pos)
- void **_M_erase_at_end** (iterator __pos)
- void **_M_fill_assign** (size_type __n, const value_type &__val)
- void _M_fill_initialize (const value_type &__value)
- void **_M_fill_insert** (iterator __pos, size_type __n, const value_type &__x)
- _Map_alloc_type **_M_get_map_allocator** () const noexcept
- template<typename _ForwardIterator>
  void **_M_insert_aux** (iterator __pos, _ForwardIterator __first, _ForwardIterator __last, size_type __n)
- iterator **_M_insert_aux** (iterator __pos, const value_type &__x)
- void **_M_insert_aux** (iterator __pos, size_type __n, const value_type &__x)
- void **_M_move_assign1** (deque &&__x, false_type)
- void **_M_move_assign1** (deque &&__x, true_type) noexcept
- void **_M_move_assign2** (deque &&__x, false_type)
- void **_M_move_assign2** (deque &&__x, true_type)
- template<typename _InputIterator, typename _Sentinel>
  void **_M_range_append** (_InputIterator __first, _Sentinel __last, size_type __n)
- void _M_range_check (size_type __n) const
- template<typename _ForwardIterator>
  void **_M_range_insert_aux** (iterator __pos, _ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- template<typename _InputIterator>
  void **_M_range_insert_aux** (iterator __pos, _InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- template<typename _InputIterator, typename _Sentinel>
  void **_M_range_prepend** (_InputIterator __first, _Sentinel __last, size_type __n)
- template<typename... _Args>
  void **_M_replace_map** (_Args &&... __args)
- bool **_M_shrink_to_fit** ()

- template<typename _InputIterator>
  void _M_range_initialize (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- template<typename _ForwardIterator>
  void _M_range_initialize (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)

- template<typename... _Args>
  void _M_push_back_aux (_Args &&... __args)
- template<typename... _Args>
  void _M_push_front_aux (_Args &&... __args)
- void _M_pop_back_aux ()
- void _M_pop_front_aux ()

- iterator _M_reserve_elements_at_front (size_type __n)
- iterator _M_reserve_elements_at_back (size_type __n)
- void _M_new_elements_at_front (size_type __new_elements)
- void _M_new_elements_at_back (size_type __new_elements)

- void _M_reserve_map_at_back (size_type __nodes_to_add=1)
- void _M_reserve_map_at_front (size_type __nodes_to_add=1)
- void _M_reallocate_map (size_type __nodes_to_add, bool __add_at_front)

Static Protected Member Functions

- static size_t **_S_check_init_len** (size_t __n, const allocator_type &__a)
- static size_type **_S_max_size** (const _Tp_alloc_type &__a) noexcept

5.381.1 Detailed Description

template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
class std::deque< _Tp, _Alloc >

A standard container using fixed-size memory allocation and constant-time manipulation of elements at either end.

Template Parameters

_Tp	Type of element.
_Alloc	Allocator type, defaults to allocator<_Tp>.

Meets the requirements of a `container`, a `reversible container`, and a `sequence`, including the `optional sequence requirements`.
In previous HP/SGI versions of deque, there was an extra template parameter so users could control the node size. This extension turned out to violate the C++ standard (it can be detected using template template parameters), and it was removed.
Here's how a deque<Tp> manages memory. Each deque has 4 members:

- Tp** _M_map
- size_t _M_map_size
- iterator _M_start, _M_finish

map_size is at least 8. map is an array of map_size pointers-to-*nodes*. (The name map has nothing to do with the std::map class, and **nodes** should not be confused with std::list's usage of *node*.)
A *node* has no specific type name as such, but it is referred to as *node* in this file. It is a simple array-of-Tp. If Tp is very large, there will be one Tp element per node (i.e., an *array* of one). For non-huge Tp's, node size is inversely related to Tp size: the larger the Tp, the fewer Tp's will fit in a node. The goal here is to keep the total size of a node relatively small and constant over different Tp's, to improve allocator efficiency.
Not every pointer in the map array will point to a node. If the initial number of elements in the deque is small, the /middle/ map pointers will be valid, and the ones at the edges will be unused. This same situation will arise as the map grows: available map pointers, if any, will be on the ends. As new nodes are created, only a subset of the map's pointers need to be copied *outward*.
Class invariants:

- For any nonsingular iterator i:
  - i.node points to a member of the map array. (Yes, you read that correctly: i.node does not actually point to a node.) The member of the map array is what actually points to the node.
  - i.first == *(i.node) (This points to the node (first Tp element).)
  - i.last == i.first + node_size
  - i.cur is a pointer in the range [i.first, i.last). NOTE: the implication of this is that i.cur is always a dereferenceable pointer, even if i is a past-the-end iterator.
- Start and Finish are always nonsingular iterators. NOTE: this means that an empty deque must have one node, a deque with <N elements (where N is the node buffer size) must have one node, a deque with N through (2N-1) elements must have two nodes, etc.
- For every node other than start.node and finish.node, every element in the node is an initialized object. If start.↩ node == finish.node, then [start.cur, finish.cur) are initialized objects, and the elements outside that range are uninitialized storage. Otherwise, [start.cur, start.last) and [finish.first, finish.cur) are initialized objects, and [start.↩ first, start.cur) and [finish.cur, finish.last) are uninitialized storage.
- [map, map + map_size) is a valid, non-empty range.
- [start.node, finish.node] is a valid range contained within [map, map + map_size).
- A pointer in the range [map, map + map_size) points to an allocated node if and only if the pointer is in the range [start.node, finish.node].

  Here's the magic: nothing in deque is **aware** of the discontiguous storage!

  The memory setup and layout occurs in the parent, _Base, and the iterator class is entirely responsible for *leaping* from one node to the next. All the implementation routines for deque itself work only through the start and finish iterators. This keeps the routines simple and sane, and we can use other standard algorithms as well.

5.381.2 Constructor & Destructor Documentation

deque() [1/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque ()  [default]
```

Creates a deque with no elements.
Referenced by insert(), and operator=().

deque() [2/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque (
            const allocator_type & __a)  [inline], [explicit]
```

Creates a deque with no elements.

Parameters

_↩ _a	An allocator object.

deque() [3/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque (
            size_type __n,
            const allocator_type & __a = allocator_type())  [inline], [explicit]
```

Creates a deque with default constructed elements.

Parameters

_↩ _n	The number of elements to initially create.
_↩ _a	An allocator.

This constructor fills the deque with *n* default constructed elements.

deque() [4/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque (
            size_type __n,
            const value_type & __value,
            const allocator_type & __a = allocator_type())  [inline]
```

Creates a deque with copies of an exemplar element.

Parameters

__n	The number of elements to initially create.
__value	An element to copy.
__a	An allocator.

This constructor fills the deque with __*n* copies of __*value*.

deque() [5/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque (
            const deque< _Tp, _Alloc > & __x)  [inline]
```

Deque copy constructor.

Parameters

_↩ _x	A deque of identical element and allocator types.

The newly-created deque uses a copy of the allocator object used by __*x* (unless the allocator traits dictate a different object).

deque() [6/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque (
            deque< _Tp, _Alloc > && )  [default]
```

Deque move constructor.
The newly-created deque contains the exact contents of the moved instance. The contents of the moved instance are a valid, but unspecified deque.

deque() [7/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque (
            const deque< _Tp, _Alloc > & __x,
            const __type_identity_t< allocator_type > & __a)  [inline]
```

Copy constructor with alternative allocator.

deque() [8/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque (
            deque< _Tp, _Alloc > && __x,
            const __type_identity_t< allocator_type > & __a)  [inline]
```

Move constructor with alternative allocator.

deque() [9/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque (
            initializer_list< value_type > __l,
            const allocator_type & __a = allocator_type())  [inline]
```

Builds a deque from an initializer list.

Parameters

_↩ _l	An initializer_list.
_↩ _a	An allocator object.

Create a deque consisting of copies of the elements in the initializer_list __l.
This will call the element type's copy constructor N times (where N is __l.size()) and do no memory reallocation.

deque() [10/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
std::deque< _Tp, _Alloc >::deque (
            _InputIterator __first,
            _InputIterator __last,
            const allocator_type & __a = allocator_type())  [inline]
```

Builds a deque from a range.

Parameters

__first	An input iterator.

Parameters

__last	An input iterator.
__a	An allocator object.

Create a deque consisting of copies of the elements from [__first, __last).
If the iterators are forward, bidirectional, or random-access, then this will call the elements' copy constructor N times (where N is distance(__first,__last)) and do no memory reallocation. But if only input iterators are used, then this will do at most 2N calls to the copy constructor, and logN memory reallocations.

~deque()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::~deque ()  [inline]
```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

5.381.3 Member Function Documentation

_M_fill_initialize()

```
template<typename _Tp, typename _Alloc>
void deque::_M_fill_initialize (
            const value_type & __value)  [protected]
```

Fills the deque with copies of value.

Parameters

__value	Initial value.

Returns

Nothing.

Precondition

_M_start and _M_finish have already been initialized, but none of the deque's elements have yet been constructed.

This function is called only when the user provides an explicit size (with or without an explicit exemplar value).
References std::_Destroy().
Referenced by std::deque< _Tp, polymorphic_allocator< _Tp > >::deque().

_M_new_elements_at_back()

```
template<typename _Tp, typename _Alloc>
void deque::_M_new_elements_at_back (
            size_type __new_elements)  [protected]
```

Memory-handling helpers for the previous internal insert functions.
References _M_reserve_map_at_back(), max_size(), and std::size().
Referenced by std::deque< _Tp, polymorphic_allocator< _Tp > >::_M_reserve_elements_at_back().

_M_new_elements_at_front()

```
template<typename _Tp, typename _Alloc>
void deque::_M_new_elements_at_front (
            size_type __new_elements)  [protected]
```

Memory-handling helpers for the previous internal insert functions.
References _M_reserve_map_at_front(), max_size(), and std::size().
Referenced by std::deque< _Tp, polymorphic_allocator< _Tp > >::_M_reserve_elements_at_front().

_M_pop_back_aux()

```
template<typename _Tp, typename _Alloc>
void deque::_M_pop_back_aux ()  [protected]
```

Helper functions for push_* and pop_*.
Referenced by std::deque< _Tp, polymorphic_allocator< _Tp > >::pop_back().

_M_pop_front_aux()

```
template<typename _Tp, typename _Alloc>
void deque::_M_pop_front_aux ()  [protected]
```

Helper functions for push_* and pop_*.
Referenced by std::deque< _Tp, polymorphic_allocator< _Tp > >::pop_front().

_M_push_back_aux()

```
template<typename _Tp, typename _Alloc>
template<typename... _Args>
void deque::_M_push_back_aux (
            _Args &&... __args)  [protected]
```

Helper functions for push_* and pop_*.
References _M_reserve_map_at_back(), std::forward(), max_size(), and std::size().
Referenced by std::deque< _Tp, polymorphic_allocator< _Tp > >::push_back().

_M_push_front_aux()

```
template<typename _Tp, typename _Alloc>
template<typename... _Args>
void deque::_M_push_front_aux (
            _Args &&... __args)  [protected]
```

Helper functions for push_* and pop_*.
References _M_reserve_map_at_front(), std::forward(), max_size(), and std::size().
Referenced by std::deque< _Tp, polymorphic_allocator< _Tp > >::push_front().

_M_range_check()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::_M_range_check (
            size_type __n) const  [inline], [protected]
```

Safety check used only from at().
Referenced by std::deque< _Tp, polymorphic_allocator< _Tp > >::at(), and std::deque< _Tp, polymorphic_allocator< _Tp > >::at().

_M_range_initialize() [1/2]

```
template<typename _Tp, typename _Alloc>
template<typename _ForwardIterator>
```

```
void deque::_M_range_initialize (
            _ForwardIterator __first,
            _ForwardIterator __last,
            std::forward_iterator_tag )  [protected]
```

Fills the deque with whatever is in [first,last).

Parameters

__first	An input iterator.
__last	An input iterator.

Returns

Nothing.

If the iterators are actually forward iterators (or better), then the memory layout can be done all at once. Else we move forward using push_back on each value from the iterator.
References std::_Destroy(), std::advance(), and std::distance().

_M_range_initialize() [2/2]

```
template<typename _Tp, typename _Alloc>
template<typename _InputIterator>
void deque::_M_range_initialize (
            _InputIterator __first,
            _InputIterator __last,
            std::input_iterator_tag )  [protected]
```

Fills the deque with whatever is in [first,last).

Parameters

__first	An input iterator.
__last	An input iterator.

Returns

Nothing.

If the iterators are actually forward iterators (or better), then the memory layout can be done all at once. Else we move forward using push_back on each value from the iterator.
References clear(), and push_back().
Referenced by std::deque< _Tp, polymorphic_allocator< _Tp > >::deque(), and std::deque< _Tp, polymorphic_allocator< _Tp > >::de

_M_reallocate_map()

```
template<typename _Tp, typename _Alloc>
void deque::_M_reallocate_map (
            size_type __nodes_to_add,
            bool __add_at_front)  [protected]
```

Memory-handling helpers for the major map.
Makes sure the _M_map has space for new nodes. Does not actually add the nodes. Can invalidate _M_map pointers. (And consequently, deque iterators.)
References std::max(), and max_size().
Referenced by std::deque< _Tp, polymorphic_allocator< _Tp > >::_M_reserve_map_at_back(), and std::deque< _Tp, polymorphic_allo

_M_reserve_elements_at_back()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::_M_reserve_elements_at_back (
            size_type __n)  [inline], [protected]
```

Memory-handling helpers for the previous internal insert functions.

_M_reserve_elements_at_front()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::_M_reserve_elements_at_front (
            size_type __n)  [inline], [protected]
```

Memory-handling helpers for the previous internal insert functions.

_M_reserve_map_at_back()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::_M_reserve_map_at_back (
            size_type __nodes_to_add = 1)  [inline], [protected]
```

Memory-handling helpers for the major map.
Makes sure the _M_map has space for new nodes. Does not actually add the nodes. Can invalidate _M_map pointers. (And consequently, deque iterators.)
Referenced by _M_new_elements_at_back(), and _M_push_back_aux().

_M_reserve_map_at_front()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::_M_reserve_map_at_front (
            size_type __nodes_to_add = 1)  [inline], [protected]
```

Memory-handling helpers for the major map.
Makes sure the _M_map has space for new nodes. Does not actually add the nodes. Can invalidate _M_map pointers. (And consequently, deque iterators.)
Referenced by _M_new_elements_at_front(), and _M_push_front_aux().

assign() [1/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
void std::deque< _Tp, _Alloc >::assign (
            _InputIterator __first,
            _InputIterator __last)  [inline]
```

Assigns a range to a deque.

Parameters

__first	An input iterator.
__last	An input iterator.

This function fills a deque with copies of the elements in the range [__first,__last).
Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned.

assign() [2/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::assign (
            initializer_list< value_type > __l)  [inline]
```

Assigns an initializer list to a deque.

Parameters

↩ _↩ ↩ _↩ *l*	An initializer_list.

This function fills a deque with copies of the elements in the initializer_list *__l*.
Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned.

assign() [3/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::assign (
            size_type __n,
            const value_type & __val)  [inline]
```

Assigns a given value to a deque.

Parameters

__n	Number of elements to be assigned.
__val	Value to be assigned.

This function fills a deque with *n* copies of the given value. Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned.

at() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::deque< _Tp, _Alloc >::at (
            size_type __n)  [inline]
```

Provides access to the data contained in the deque.

Parameters

_↩ *_n*	The index of the element for which data should be accessed.

Returns

Read/write reference to data.

Exceptions

std::out_of_range	If *__n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the deque. The function throws out_of_range if the check fails.

at() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::deque< _Tp, _Alloc >::at (
            size_type __n) const  [inline]
```

Provides access to the data contained in the deque.

Parameters

__n	The index of the element for which data should be accessed.

Returns

Read-only (constant) reference to data.

Exceptions

std::out_of_range	If *__n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the deque. The function throws out_of_range if the check fails.

back() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::deque< _Tp, _Alloc >::back () const  [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) reference to the data at the last element of the deque.

back() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::deque< _Tp, _Alloc >::back ()  [inline], [nodiscard], [noexcept]
```

Returns a read/write reference to the data at the last element of the deque.

begin() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::deque< _Tp, _Alloc >::begin () const  [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the deque. Iteration is done in ordinary element order.

begin() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::begin ()  [inline], [nodiscard], [noexcept]
```

Returns a read/write iterator that points to the first element in the deque. Iteration is done in ordinary element order.
Referenced by std::deque< _Tp, polymorphic_allocator< _Tp > >::deque(), std::deque< _Tp, polymorphic_allocator< _Tp > >::deque
std::deque< _Tp, polymorphic_allocator< _Tp > >::~deque(), std::deque< _Tp, polymorphic_allocator< _Tp > >::clear(),

std::deque< _Tp, polymorphic_allocator< _Tp > >::front(), std::deque< _Tp, polymorphic_allocator< _Tp > >::front(), std::deque< _Tp, polymorphic_allocator< _Tp > >::insert(), std::deque< _Tp, polymorphic_allocator< _Tp > >::insert(), std::deque< _Tp, polymorphic_allocator< _Tp > >::insert(), std::operator<=>(), and operator=().

cbegin()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::deque< _Tp, _Alloc >::cbegin () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the deque. Iteration is done in ordinary element order.

Referenced by std::deque< _Tp, polymorphic_allocator< _Tp > >::insert(), std::deque< _Tp, polymorphic_allocator< _Tp > >::insert() and std::deque< _Tp, polymorphic_allocator< _Tp > >::insert().

cend()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::deque< _Tp, _Alloc >::cend () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the deque. Iteration is done in ordinary element order.

clear()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::clear ()  [inline], [noexcept]
```

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Referenced by _M_range_initialize().

crbegin()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::deque< _Tp, _Alloc >::crbegin () const  [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the deque. Iteration is done in reverse element order.

crend()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::deque< _Tp, _Alloc >::crend () const  [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first element in the deque. Iteration is done in reverse element order.

emplace()

```
template<typename _Tp, typename _Alloc>
template<typename...  _Args>
deque< _Tp, _Alloc >::iterator deque::emplace (
            const_iterator __position,
            _Args &&...  __args)
```

Inserts an object in deque before specified iterator.

Parameters

__position	A const_iterator into the deque.
__args	Arguments.

Returns

An iterator that points to the inserted data.

This function will insert an object of type T constructed with T(std::forward<Args>(args)...) before the specified location.
References std::forward().
Referenced by std::deque< _Tp, polymorphic_allocator< _Tp > >::insert().

empty()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
bool std::deque< _Tp, _Alloc >::empty () const  [inline], [nodiscard], [noexcept]
```

Returns true if the deque is empty. (Thus begin() would equal end().)

end() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::deque< _Tp, _Alloc >::end () const  [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the deque. Iteration is done in ordinary element order.

end() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::end ()  [inline], [nodiscard], [noexcept]
```

Returns a read/write iterator that points one past the last element in the deque. Iteration is done in ordinary element order.
Referenced by std::deque< _Tp, polymorphic_allocator< _Tp > >::deque(), std::deque< _Tp, polymorphic_allocator< _Tp > >::deque
std::deque< _Tp, polymorphic_allocator< _Tp > >::~deque(), std::deque< _Tp, polymorphic_allocator< _Tp > >::back(),
std::deque< _Tp, polymorphic_allocator< _Tp > >::back(), std::operator<=>(), and operator=().

erase() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::erase (
            const_iterator __first,
            const_iterator __last)  [inline]
```

Remove a range of elements.

Parameters

__first	Iterator pointing to the first element to be erased.
__last	Iterator pointing to one past the last element to be erased.

Returns

An iterator pointing to the element pointed to by *last* prior to erasing (or end()).

This function will erase the elements in the range [__first,__last) and shorten the deque accordingly.
The user is cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::erase (
            const_iterator __position)  [inline]
```

Remove element at given position.

Parameters

__position	Iterator pointing to element to be erased.

Returns

An iterator pointing to the next element (or end()).

This function will erase the element at the given position and thus shorten the deque by one.
The user is cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

front() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::deque< _Tp, _Alloc >::front () const  [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) reference to the data at the first element of the deque.

front() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::deque< _Tp, _Alloc >::front ()  [inline], [nodiscard], [noexcept]
```

Returns a read/write reference to the data at the first element of the deque.

get_allocator()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
allocator_type std::deque< _Tp, _Alloc >::get_allocator () const  [inline], [nodiscard], [noexcept]
```

Get a copy of the memory allocation object.
Referenced by operator=().

insert() [1/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::insert (
            const_iterator __p,
            initializer_list< value_type > __l)  [inline]
```

Inserts an initializer list into the deque.

Parameters

_↩ _p	An iterator into the deque.
_↩ _l	An initializer_list.

Returns

An iterator that points to the inserted data.

This function will insert copies of the data in the initializer_list __*l* into the deque before the location specified by __*p*. This is known as *list insert*.

insert() [2/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
iterator std::deque< _Tp, _Alloc >::insert (
            const_iterator __position,
            _InputIterator __first,
            _InputIterator __last)  [inline]
```

Inserts a range into the deque.

Parameters

__position	A const_iterator into the deque.
__first	An input iterator.
__last	An input iterator.

Returns

An iterator that points to the inserted data.

This function will insert copies of the data in the range [__first,__last) into the deque before the location specified by __*position*. This is known as *range insert*.

insert() [3/5]

```
template<typename _Tp, typename _Alloc>
deque< _Tp, _Alloc >::iterator deque::insert (
            const_iterator __position,
            const value_type & __x)
```

Inserts given value into deque before specified iterator.

Parameters

__position	A const_iterator into the deque.
__x	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location.
References deque(), insert(), push_back(), and push_front().
Referenced by insert().

insert() [4/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::insert (
            const_iterator __position,
            size_type __n,
            const value_type & __x)  [inline]
```

Inserts a number of copies of given data into the deque.

Parameters

__position	A const_iterator into the deque.
__n	Number of elements to be inserted.
__x	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a specified number of copies of the given data before the location specified by *__position*.

insert() [5/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::insert (
            const_iterator __position,
            value_type && __x)  [inline]
```

Inserts given rvalue into deque before specified iterator.

Parameters

__position	A const_iterator into the deque.
__x	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location.

max_size()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::deque< _Tp, _Alloc >::max_size () const  [inline], [nodiscard], [noexcept]
```

Returns the size() of the largest possible deque.

Referenced by _M_new_elements_at_back(), _M_new_elements_at_front(), _M_push_back_aux(), _M_push_front_aux(), and _M_reallocate_map().

operator=() [1/3]

```
template<typename _Tp, typename _Alloc>
deque< _Tp, _Alloc > & deque::operator= (
            const deque< _Tp, _Alloc > & __x)
```

Deque assignment operator.

Parameters

_↩ _x	A deque of identical element and allocator types.

All the elements of *x* are copied.
The newly-created deque uses a copy of the allocator object used by *__x* (unless the allocator traits dictate a different object).
References deque(), std::__addressof(), begin(), end(), get_allocator(), size(), and std::size().

operator=() [2/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
deque & std::deque< _Tp, _Alloc >::operator= (
            deque< _Tp, _Alloc > && __x)  [inline], [noexcept]
```

Deque move assignment operator.

Parameters

_↩ _x	A deque of identical element and allocator types.

The contents of *__x* are moved into this deque (without copying, if the allocators permit it). *__x* is a valid, but unspecified deque.

operator=() [3/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
deque & std::deque< _Tp, _Alloc >::operator= (
            initializer_list< value_type > __l)  [inline]
```

Assigns an initializer list to a deque.

Parameters

↩ _↩ ↩ _↩ *l*	An initializer_list.

This function fills a deque with copies of the elements in the initializer_list *__l*.
Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned.

operator[]() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::deque< _Tp, _Alloc >::operator[] (
            size_type __n) const  [inline], [nodiscard], [noexcept]
```

Subscript access to the data contained in the deque.

Parameters

_↩ _n	The index of the element for which data should be accessed.

Returns

Read-only (constant) reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out_of_range lookups are not defined. (For checked lookups see at().)

operator[]() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::deque< _Tp, _Alloc >::operator[] (
            size_type __n)  [inline], [nodiscard], [noexcept]
```

Subscript access to the data contained in the deque.

Parameters

_↩ _n	The index of the element for which data should be accessed.

Returns

Read/write reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out_of_range lookups are not defined. (For checked lookups see at().)

pop_back()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::pop_back ()  [inline], [noexcept]
```

Removes last element.
This is a typical stack operation. It shrinks the deque by one.
Note that no data is returned, and if the last element's data is needed, it should be retrieved before pop_back() is called.

pop_front()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::pop_front ()  [inline], [noexcept]
```

Removes first element.
This is a typical stack operation. It shrinks the deque by one.
Note that no data is returned, and if the first element's data is needed, it should be retrieved before pop_front() is called.

push_back()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::push_back (
            const value_type & __x)  [inline]
```

Add data to the end of the deque.

Parameters

_↩ _x	Data to be added.

This is a typical stack operation. The function creates an element at the end of the deque and assigns the given data to it. Due to the nature of a deque this operation can be done in constant time.
Referenced by _M_range_initialize(), and insert().

push_front()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::push_front (
            const value_type & __x)  [inline]
```

Add data to the front of the deque.

Parameters

_↩ _x	Data to be added.

This is a typical stack operation. The function creates an element at the front of the deque and assigns the given data to it. Due to the nature of a deque this operation can be done in constant time.
Referenced by insert().

rbegin() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::deque< _Tp, _Alloc >::rbegin () const  [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the deque. Iteration is done in reverse element order.

rbegin() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::deque< _Tp, _Alloc >::rbegin ()  [inline], [nodiscard], [noexcept]
```

Returns a read/write reverse iterator that points to the last element in the deque. Iteration is done in reverse element order.

rend() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::deque< _Tp, _Alloc >::rend () const  [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first element in the deque. Iteration is done in reverse element order.

rend() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::deque< _Tp, _Alloc >::rend ()  [inline], [nodiscard], [noexcept]
```

Returns a read/write reverse iterator that points to one before the first element in the deque. Iteration is done in reverse element order.

resize() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::resize (
            size_type __new_size)  [inline]
```

Resizes the deque to the specified number of elements.

Parameters

__new_size	Number of elements the deque should contain.

This function will resize the deque to the specified number of elements. If the number is smaller than the deque's current size the deque is truncated, otherwise default constructed elements are appended.

resize() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::resize (
            size_type __new_size,
            const value_type & __x)  [inline]
```

Resizes the deque to the specified number of elements.

Parameters

__new_size	Number of elements the deque should contain.
__x	Data with which new elements should be populated.

This function will resize the deque to the specified number of elements. If the number is smaller than the deque's current size the deque is truncated, otherwise the deque is extended and new elements are populated with given data.

shrink_to_fit()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::shrink_to_fit ()  [inline], [noexcept]
```

A non-binding request to reduce memory use.

size()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::deque< _Tp, _Alloc >::size () const  [inline], [nodiscard], [noexcept]
```

Returns the number of elements in the deque.

Referenced by std::deque< _Tp, polymorphic_allocator< _Tp > >::_M_range_check(), std::operator<=>(), operator=(), std::deque< _Tp, polymorphic_allocator< _Tp > >::resize(), and std::deque< _Tp, polymorphic_allocator< _Tp > >::resize().

swap()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::swap (
            deque< _Tp, _Alloc > & __x)  [inline], [noexcept]
```

Swaps data with another deque.

Parameters

_↩ _x	A deque of the same element and allocator types.

This exchanges the elements between two deques in constant time. (Four pointers, so it should be quite fast.) Note that the global std::swap() function is specialized such that std::swap(d1,d2) will feed to this function.
Whether the allocators are swapped depends on the allocator traits.
The documentation for this class was generated from the following files:

- stl_deque.h
- deque.tcc

5.382 std::destroying_delete_t Struct Reference

`#include <new>`

5.382.1 Detailed Description

Tag type used to declare a class-specific operator delete that can invoke the destructor before deallocating the memory.
The documentation for this struct was generated from the following file:

- new

5.383 std::tr2::direct_bases< _Tp > Struct Template Reference

`#include <type_traits>`

Public Types

- typedef __reflection_typelist< __direct_bases(_Tp)... > **type**

5.383.1 Detailed Description

template<typename _Tp>
struct std::tr2::direct_bases< _Tp >

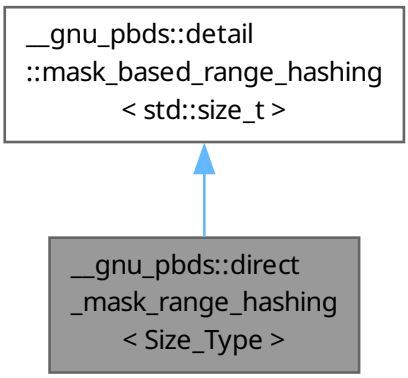
Enumerate all the direct base classes of a class. Form of a typelist.
The documentation for this struct was generated from the following file:

- tr2/type_traits

5.384 __gnu_pbds::direct_mask_range_hashing< Size_Type > Class Template Reference

`#include <hash_policy.hpp>`

Inheritance diagram for __gnu_pbds::direct_mask_range_hashing< Size_Type >:



Public Types

- typedef Size_Type **size_type**

Public Member Functions

- void **swap** (direct_mask_range_hashing< Size_Type > &other)

Protected Member Functions

- void **notify_resized** (size_type size)
- size_type operator() (size_type hash) const
- size_type **range_hash** (size_type hash) const
- void **swap** (mask_based_range_hashing &other)

5.384.1 Detailed Description

template<typename Size_Type = std::size_t>
class __gnu_pbds::direct_mask_range_hashing< Size_Type >

A mask range-hashing class (uses a bitmask).

5.384.2 Member Function Documentation

operator()()

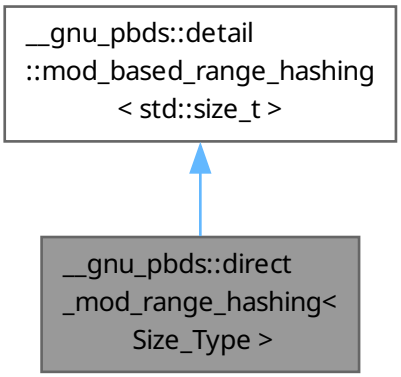
```
template<typename Size_Type>
direct_mask_range_hashing< Size_Type >::size_type __gnu_pbds::direct_mask_range_hashing< Size_↩
Type >::operator() (
            size_type hash) const  [inline], [protected]
```

Transforms the __hash value hash into a ranged-hash value (using a bit-mask).
The documentation for this class was generated from the following file:

- hash_policy.hpp

5.385 __gnu_pbds::direct_mod_range_hashing< Size_Type > Class Template Reference

`#include <hash_policy.hpp>`
Inheritance diagram for __gnu_pbds::direct_mod_range_hashing< Size_Type >:



Public Types

- typedef Size_Type **size_type**

Public Member Functions

- void **swap** (direct_mod_range_hashing< Size_Type > &other)

Protected Member Functions

- void **notify_resized** (size_type s)
- void **notify_resized** (size_type size)
- size_type operator() (size_type hash) const
- size_type **range_hash** (size_type s) const
- void **swap** (mod_based_range_hashing &other)

5.385.1 Detailed Description

template<typename Size_Type = std::size_t>
class __gnu_pbds::direct_mod_range_hashing< Size_Type >

A mod range-hashing class (uses the modulo function).

5.385.2 Member Function Documentation

operator()()

```
template<typename Size_Type>
direct_mod_range_hashing< Size_Type >::size_type __gnu_pbds::direct_mod_range_hashing< Size_Type
>::operator() (
            size_type hash) const  [inline], [protected]
```

Transforms the __hash value hash into a ranged-hash value (using a modulo operation).
The documentation for this class was generated from the following file:

- hash_policy.hpp

5.386 std::filesystem::directory_entry Class Reference

`#include <filesystem>`

Public Member Functions

- **directory_entry** (const directory_entry &)=default
- **directory_entry** (const filesystem::path &__p)
- **directory_entry** (const filesystem::path &__p, error_code &__ec)
- **directory_entry** (directory_entry &&) noexcept=default
- void **assign** (const filesystem::path &__p)
- void **assign** (const filesystem::path &__p, error_code &__ec)
- bool **exists** () const
- bool **exists** (error_code &__ec) const noexcept
- uintmax_t **file_size** () const
- uintmax_t **file_size** (error_code &__ec) const noexcept
- uintmax_t **hard_link_count** () const
- uintmax_t **hard_link_count** (error_code &__ec) const noexcept
- bool **is_block_file** () const
- bool **is_block_file** (error_code &__ec) const noexcept
- bool **is_character_file** () const
- bool **is_character_file** (error_code &__ec) const noexcept
- bool **is_directory** () const
- bool **is_directory** (error_code &__ec) const noexcept
- bool **is_fifo** () const
- bool **is_fifo** (error_code &__ec) const noexcept
- bool **is_other** () const
- bool **is_other** (error_code &__ec) const noexcept
- bool **is_regular_file** () const
- bool **is_regular_file** (error_code &__ec) const noexcept
- bool **is_socket** () const
- bool **is_socket** (error_code &__ec) const noexcept
- bool **is_symlink** () const
- bool **is_symlink** (error_code &__ec) const noexcept
- file_time_type **last_write_time** () const
- file_time_type **last_write_time** (error_code &__ec) const noexcept
- **operator const filesystem::path &** () const noexcept
- strong_ordering **operator<=>** (const directory_entry &__rhs) const noexcept
- directory_entry & **operator=** (const directory_entry &)=default
- directory_entry & **operator=** (directory_entry &&) noexcept=default
- bool **operator==** (const directory_entry &__rhs) const noexcept
- const filesystem::path & **path** () const noexcept
- void **refresh** ()
- void **refresh** (error_code &__ec) noexcept
- void **replace_filename** (const filesystem::path &__p)
- void **replace_filename** (const filesystem::path &__p, error_code &__ec)
- file_status **status** () const
- file_status **status** (error_code &__ec) const noexcept
- file_status **symlink_status** () const
- file_status **symlink_status** (error_code &__ec) const noexcept

Friends

- struct **_Dir**
- class **directory_iterator**
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator<<** (basic_ostream< _CharT, _Traits > &__os, const directory_entry &__d)
- class **recursive_directory_iterator**

5.386.1 Detailed Description

The value type used by directory iterators.

Since

C++17

The documentation for this class was generated from the following file:

- bits/fs_dir.h

5.387 std::filesystem::directory_iterator Class Reference

`#include <filesystem>`

Public Types

- typedef ptrdiff_t **difference_type**
- typedef input_iterator_tag **iterator_category**
- typedef const directory_entry * **pointer**
- typedef const directory_entry & **reference**
- typedef directory_entry **value_type**

Public Member Functions

- **directory_iterator** (const directory_iterator &__rhs)=default
- **directory_iterator** (const path &__p)
- **directory_iterator** (const path &__p, directory_options __options)
- **directory_iterator** (const path &__p, directory_options __options, error_code &__ec)
- **directory_iterator** (const path &__p, error_code &__ec)
- **directory_iterator** (directory_iterator &&__rhs) noexcept=default
- directory_iterator & **increment** (error_code &__ec)
- const directory_entry & **operator*** () const noexcept
- directory_iterator & **operator++** ()
- __directory_iterator_proxy **operator++** (int)
- const directory_entry * **operator->** () const noexcept
- directory_iterator & **operator=** (const directory_iterator &__rhs)=default
- directory_iterator & **operator=** (directory_iterator &&__rhs) noexcept=default
- bool **operator==** (default_sentinel_t) const noexcept

Friends

- bool **operator==** (const directory_iterator &__lhs, const directory_iterator &__rhs) noexcept
- class **recursive_directory_iterator**

Related Symbols

(Note that these are not member symbols.)

- directory_iterator begin (directory_iterator __iter) noexcept
- directory_iterator end (directory_iterator) noexcept

- recursive_directory_iterator begin (recursive_directory_iterator __iter) noexcept
- recursive_directory_iterator end (recursive_directory_iterator) noexcept

5.387.1 Detailed Description

Iterator type for traversing the entries in a single directory.

Since

C++17

The documentation for this class was generated from the following file:

- bits/fs_dir.h

5.388 std::discard_block_engine< _RandomNumberEngine, __p, __r > Class Template Reference

`#include <random>`

Public Types

- template<typename _Sseq>
  using **_If_seed_seq**
- typedef _RandomNumberEngine::result_type result_type

Public Member Functions

- discard_block_engine ()
- discard_block_engine (_RandomNumberEngine &&__rng)
- template<typename _Sseq, typename = _If_seed_seq<_Sseq>>
  discard_block_engine (_Sseq &__q)
- discard_block_engine (const _RandomNumberEngine &__rng)
- discard_block_engine (result_type __s)
- const _RandomNumberEngine & base () const noexcept
- void discard (unsigned long long __z)
- result_type operator() ()
- void seed ()
- template<typename _Sseq>
  _If_seed_seq< _Sseq > seed (_Sseq &__q)
- void seed (result_type __s)

Static Public Member Functions

- static constexpr result_type max ()
- static constexpr result_type min ()

Static Public Attributes

- static constexpr size_t **block_size**
- static constexpr size_t **used_block**

Friends

- template<typename _RandomNumberEngine1, size_t __p1, size_t __r1, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::discard_block_engine< _RandomNumberEngine1, __p1, __r1 > &__x)
- bool operator== (const discard_block_engine &__lhs, const discard_block_engine &__rhs)
- template<typename _RandomNumberEngine1, size_t __p1, size_t __r1, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::discard_block_engine< _RandomNumberEngine1, __p1, __r1 > &__x)

5.388.1 Detailed Description

template<typename _RandomNumberEngine, size_t __p, size_t __r>
class std::discard_block_engine< _RandomNumberEngine, __p, __r >

Produces random numbers from some base engine by discarding blocks of data.

Precondition

$0 \leq r \leq p$

Since

C++11

5.388.2 Member Typedef Documentation

result_type

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
typedef _RandomNumberEngine::result_type std::discard_block_engine< _RandomNumberEngine, __p, __r
>::result_type
```

The type of the generated random value.

5.388.3 Constructor & Destructor Documentation

discard_block_engine() [1/5]

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
std::discard_block_engine< _RandomNumberEngine, __p, __r >::discard_block_engine ()  [inline]
```

Constructs a default discard_block_engine engine.
The underlying engine is default constructed as well.

discard_block_engine() [2/5]

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
std::discard_block_engine< _RandomNumberEngine, __p, __r >::discard_block_engine (
            const _RandomNumberEngine & __rng)  [inline], [explicit]
```

Copy constructs a discard_block_engine engine.
Copies an existing base class random number generator.

Parameters

__rng	An existing (base class) engine object.

discard_block_engine() [3/5]

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
std::discard_block_engine< _RandomNumberEngine, __p, __r >::discard_block_engine (
            _RandomNumberEngine && __rng)  [inline], [explicit]
```

Move constructs a discard_block_engine engine.
Copies an existing base class random number generator.

Parameters

__rng	An existing (base class) engine object.

discard_block_engine() [4/5]

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
std::discard_block_engine< _RandomNumberEngine, __p, __r >::discard_block_engine (
            result_type __s)  [inline], [explicit]
```

Seed constructs a discard_block_engine engine.
Constructs the underlying generator engine seeded with __s.

Parameters

_↩ _s	A seed value for the base class engine.

discard_block_engine() [5/5]

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
template<typename _Sseq, typename = _If_seed_seq<_Sseq>>
std::discard_block_engine< _RandomNumberEngine, __p, __r >::discard_block_engine (
            _Sseq & __q)  [inline], [explicit]
```

Generator construct a discard_block_engine engine.

Parameters

_↩ _q	A seed sequence.

5.388.4 Member Function Documentation

base()

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
const _RandomNumberEngine & std::discard_block_engine< _RandomNumberEngine, __p, __r >::base ()
const  [inline], [noexcept]
```

Gets a const reference to the underlying generator engine object.

discard()

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
void std::discard_block_engine< _RandomNumberEngine, __p, __r >::discard (
            unsigned long long __z)  [inline]
```

Discard a sequence of random numbers.

max()

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
static constexpr result_type std::discard_block_engine< _RandomNumberEngine, __p, __r >::max ()
[inline], [static], [constexpr]
```

Gets the maximum value in the generated random number range.

min()

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
static constexpr result_type std::discard_block_engine< _RandomNumberEngine, __p, __r >::min ()
[inline], [static], [constexpr]
```

Gets the minimum value in the generated random number range.

operator()()

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
discard_block_engine< _RandomNumberEngine, __p, __r >::result_type std::discard_block_engine< _↩
RandomNumberEngine, __p, __r >::operator() ()
```

Gets the next value in the generated random number sequence.

seed() [1/3]

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
void std::discard_block_engine< _RandomNumberEngine, __p, __r >::seed ()  [inline]
```

Reseeds the discard_block_engine object with the default seed for the underlying base class generator engine.

seed() [2/3]

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
template<typename _Sseq>
_If_seed_seq< _Sseq > std::discard_block_engine< _RandomNumberEngine, __p, __r >::seed (
            _Sseq & __q)  [inline]
```

Reseeds the discard_block_engine object with the given seed sequence.

Parameters

_↩ _q	A seed generator function.

seed() [3/3]

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
void std::discard_block_engine< _RandomNumberEngine, __p, __r >::seed (
            result_type __s)  [inline]
```

Reseeds the discard_block_engine object with the default seed for the underlying base class generator engine.

5.388.5 Friends And Related Symbol Documentation

operator<<

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
template<typename _RandomNumberEngine1, size_t __p1, size_t __r1, typename _CharT, typename _↩
Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::discard_block_engine< _RandomNumberEngine1, __p1, __r1 > & __x)  [friend]
```

Inserts the current state of a discard_block_engine random number generator engine __x into the output stream __os.

Parameters

__os	An output stream.
__x	A discard_block_engine random number generator engine.

Returns

The output stream with the state of __x inserted or in an error state.

operator==

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
bool operator== (
            const discard_block_engine< _RandomNumberEngine, __p, __r > & __lhs,
            const discard_block_engine< _RandomNumberEngine, __p, __r > & __rhs)  [friend]
```

Compares two discard_block_engine random number generator objects of the same type for equality.

Parameters

__lhs	A discard_block_engine random number generator object.
__rhs	Another discard_block_engine random number generator object.

Returns

true if the infinite sequences of generated values would be equal, false otherwise.

operator>>

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
template<typename _RandomNumberEngine1, size_t __p1, size_t __r1, typename _CharT, typename _↩
Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            std::discard_block_engine< _RandomNumberEngine1, __p1, __r1 > & __x)  [friend]
```

Extracts the current state of a % subtract_with_carry_engine random number generator engine __x from the input stream __is.

Parameters

_↩ _is	An input stream.

_↩ _x	A discard_block_engine random number generator engine.

Returns

The input stream with the state of __x extracted or in an error state.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

5.389 std::discrete_distribution< _IntType > Class Template Reference

`#include <random>`

Classes

- struct param_type

Public Types

- typedef _IntType result_type

Public Member Functions

- template<typename _InputIterator>
  discrete_distribution (_InputIterator __wbegin, _InputIterator __wend)
- **discrete_distribution** (const param_type &__p)
- **discrete_distribution** (initializer_list< double > __wl)
- template<typename _Func>
  discrete_distribution (size_t __nw, double __xmin, double __xmax, _Func __fw)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- std::vector< double > probabilities () const
- void reset ()

Friends

- template<typename _IntType1, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::discrete_distribution< _IntType1 > &__x)
- bool operator== (const discrete_distribution &__d1, const discrete_distribution &__d2)
- template<typename _IntType1, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::discrete_distribution< _IntType1 > &__x)

5.389.1 Detailed Description

template<typename _IntType = int>
class std::discrete_distribution< _IntType >

A discrete_distribution random number distribution.
This distribution produces random numbers $i, 0 \leq i < n$, distributed according to the probability mass function $p(i|p_0, ..., p_{n-1}) = p_i$.

Since

C++11

5.389.2 Member Typedef Documentation

result_type

```
template<typename _IntType = int>
typedef _IntType std::discrete_distribution< _IntType >::result_type
```

The type of the range of the distribution.

5.389.3 Member Function Documentation

max()

```
template<typename _IntType = int>
result_type std::discrete_distribution< _IntType >::max () const  [inline]
```

Returns the least upper bound value of the distribution.

min()

```
template<typename _IntType = int>
result_type std::discrete_distribution< _IntType >::min () const  [inline]
```

Returns the greatest lower bound value of the distribution.

operator()()

```
template<typename _IntType = int>
template<typename _UniformRandomNumberGenerator>
result_type std::discrete_distribution< _IntType >::operator() (
            _UniformRandomNumberGenerator & __urng)  [inline]
```

Generating functions.
Referenced by operator()().

param() [1/2]

```
template<typename _IntType = int>
param_type std::discrete_distribution< _IntType >::param () const  [inline]
```

Returns the parameter set of the distribution.

param() [2/2]

```
template<typename _IntType = int>
void std::discrete_distribution< _IntType >::param (
            const param_type & __param)  [inline]
```

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

probabilities()

```
template<typename _IntType = int>
std::vector< double > std::discrete_distribution< _IntType >::probabilities () const  [inline]
```

Returns the probabilities of the distribution.

reset()

```
template<typename _IntType = int>
void std::discrete_distribution< _IntType >::reset ()  [inline]
```

Resets the distribution state.

5.389.4 Friends And Related Symbol Documentation

operator<<

```
template<typename _IntType = int>
template<typename _IntType1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::discrete_distribution< _IntType1 > & __x)  [friend]
```

Inserts a discrete_distribution random number distribution __x into the output stream __os.

Parameters

__os	An output stream.
__x	A discrete_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

operator==

```
template<typename _IntType = int>
bool operator== (
             const discrete_distribution< _IntType > & __d1,
             const discrete_distribution< _IntType > & __d2)  [friend]
```

Return true if two discrete distributions have the same parameters.

operator>>

```
template<typename _IntType = int>
template<typename _IntType1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
             std::basic_istream< _CharT, _Traits > & __is,
             std::discrete_distribution< _IntType1 > & __x)  [friend]
```

Extracts a discrete_distribution random number distribution __x from the input stream __is.

Parameters

_↩ _is	An input stream.
_↩ _x	A discrete_distribution random number generator engine.

Returns

The input stream with __x extracted or in an error state.

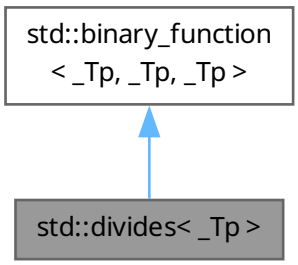
The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

5.390 std::divides< _Tp > Struct Template Reference

`#include <stl_function.h>`

Inheritance diagram for std::divides< _Tp >:



Public Types

- typedef _Tp first_argument_type
- typedef _Tp result_type
- typedef _Tp second_argument_type

Public Member Functions

- constexpr _Tp **operator()** (const _Tp &__x, const _Tp &__y) const

5.390.1 Detailed Description

template<typename _Tp>
struct std::divides< _Tp >

One of the math functors.

5.390.2 Member Typedef Documentation

first_argument_type

```
typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::first_argument_type  [inherited]
```

`first_argument_type` is the type of the first argument

result_type

```
typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::result_type  [inherited]
```

`result_type` is the return type

second_argument_type

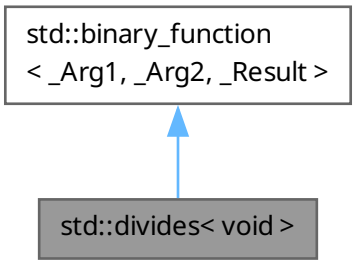
```
typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::second_argument_type  [inherited]
```

`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

5.391 std::divides< void > Struct Reference

`#include <stl_function.h>`
Inheritance diagram for std::divides< void >:



Public Types

- typedef void first_argument_type
- typedef __is_transparent **is_transparent**
- typedef void result_type
- typedef void second_argument_type

Public Member Functions

- template<typename _Tp, typename _Up>
  constexpr auto **operator()** (_Tp &&__t, _Up &&__u) const noexcept(noexcept(std::forward< _Tp >(__t)/std::forward< _Up >(__u))) -> decltype(std::forward< _Tp >(__t)/std::forward< _Up >(__u))
- constexpr void **operator()** (const void &__x, const void &__y) const

5.391.1 Detailed Description

One of the math functors.

5.391.2 Member Typedef Documentation

first_argument_type

```
typedef void std::binary_function< void, void, void >::first_argument_type
```

`first_argument_type` is the type of the first argument

result_type

```
typedef void std::binary_function< void, void, void >::result_type
```

`result_type` is the return type

second_argument_type

```
typedef void std::binary_function< void, void, void >::second_argument_type
```

`second_argument_type` is the type of the second argument

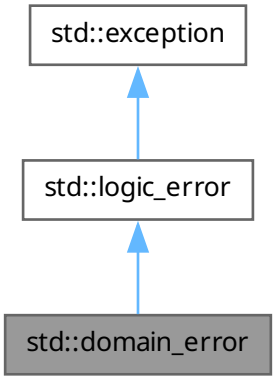
The documentation for this struct was generated from the following file:

- stl_function.h

5.392 std::domain_error Class Reference

```
#include <stdexcept>
```

Inheritance diagram for std::domain_error:



Public Member Functions

- **domain_error** (const char *)
- **domain_error** (const domain_error &)=default
- **domain_error** (const string &__arg)
- **domain_error** (domain_error &&)=default
- domain_error & **operator=** (const domain_error &)=default
- domain_error & **operator=** (domain_error &&)=default
- virtual const char * what () const noexcept

5.392.1 Detailed Description

Thrown by the library, or by you, to report domain errors (domain in the mathematical sense).

5.392.2 Member Function Documentation

what()

```
virtual const char * std::logic_error::what () const  [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).
Reimplemented from std::exception.
Reimplemented in std::future_error.
The documentation for this class was generated from the following file:

- stdexcept

5.393 __gnu_pbds::detail::dumnode_const_iterator< Key, Data, _Alloc > Struct Template Reference

```
#include <null_node_metadata.hpp>
```

Public Types

- typedef const_iterator **const_reference**
- typedef const_reference **reference**
- typedef const_iterator **value_type**

5.393.1 Detailed Description

template<typename Key, typename Data, typename _Alloc>
struct __gnu_pbds::detail::dumnode_const_iterator< Key, Data, _Alloc >

Constant node iterator.
The documentation for this struct was generated from the following file:

- null_node_metadata.hpp

5.394 std::chrono::duration< _Rep, _Period > Class Template Reference

Public Types

- using **period**
- using **rep**

Public Member Functions

- template<typename _Rep2, typename = _Require< is_convertible<const _Rep2&, rep>, __or_<__is_float<rep>, __not_<__is_float<_Rep2>>>>>
  constexpr **duration** (const _Rep2 &__rep)
- **duration** (const duration &)=default
- template<typename _Rep2, typename _Period2, typename = _Require< is_convertible<const _Rep2&, rep>, __or_<__is_float<rep>, __and_<__is_harmonic<_Period2>, __not_<__is_float<_Rep2>>>>>>
  constexpr **duration** (const duration< _Rep2, _Period2 > &__d)
- constexpr rep **count** () const
- template<typename _Rep2 = rep>
  constexpr __enable_if_t<!treat_as_floating_point< _Rep2 >::value, duration & > **operator%=** (const duration &__d)
- template<typename _Rep2 = rep>
  constexpr __enable_if_t<!treat_as_floating_point< _Rep2 >::value, duration & > **operator%=** (const rep &__rhs)
- constexpr duration & **operator*=** (const rep &__rhs)
- constexpr duration< typename common_type< rep >::type, period > **operator+** () const
- constexpr duration & **operator++** ()
- constexpr duration **operator++** (int)
- constexpr duration & **operator+=** (const duration &__d)
- constexpr duration< typename common_type< rep >::type, period > **operator-** () const
- constexpr duration & **operator--** ()
- constexpr duration **operator--** (int)
- constexpr duration & **operator-=** (const duration &__d)
- constexpr duration & **operator/=** (const rep &__rhs)
- duration & **operator=** (const duration &)=default

Static Public Member Functions

- static constexpr duration **max** () noexcept
- static constexpr duration **min** () noexcept
- static constexpr duration **zero** () noexcept

Related Symbols

(Note that these are not member symbols.)

- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type operator+ (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)

- template<typename _Rep1, typename _Period, typename _Rep2>
  constexpr duration< __common_rep_t< _Rep1, _Rep2 >, _Period > operator* (const duration< _Rep1, _Period > &__d, const _Rep2 &__s)

- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr bool operator== (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)

5.394.1 Detailed Description

template<typename _Rep, typename _Period>
class std::chrono::duration< _Rep, _Period >

`chrono::duration` represents a distance between two points in time
The documentation for this class was generated from the following file:

- chrono.h

5.395 std::chrono::duration_values< _Rep > Struct Template Reference

`#include <chrono.h>`

Static Public Member Functions

- static constexpr _Rep **max** () noexcept
- static constexpr _Rep **min** () noexcept
- static constexpr _Rep **zero** () noexcept

5.395.1 Detailed Description

template<typename _Rep>
struct std::chrono::duration_values< _Rep >

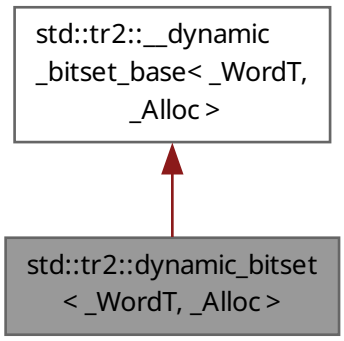
duration_values
The documentation for this struct was generated from the following file:

- chrono.h

5.396 std::tr2::dynamic_bitset< _WordT, _Alloc > Class Template Reference

`#include <dynamic_bitset>`

Inheritance diagram for std::tr2::dynamic_bitset< _WordT, _Alloc >:



Classes

- class reference

Public Types

- typedef __dynamic_bitset_base< _WordT, _Alloc > **_Base**
- typedef _Alloc **allocator_type**
- typedef _WordT **block_type**
- typedef bool **const_reference**
- typedef size_t **size_type**

Public Member Functions

- dynamic_bitset ()=default
- dynamic_bitset (const allocator_type &__alloc)
- dynamic_bitset (const char *__str, const allocator_type &__alloc=allocator_type())
- dynamic_bitset (const dynamic_bitset &)=default
- template<typename _CharT, typename _Traits, typename _Alloc1>
  dynamic_bitset (const std::basic_string< _CharT, _Traits, _Alloc1 > &__str, typename basic_string< _CharT, _Traits, _Alloc1 >::size_type __pos=0, typename basic_string< _CharT, _Traits, _Alloc1 >::size_type __n=std::basic_string< _CharT, _Traits, _Alloc1 >::npos, _CharT __zero=_CharT('0'), _CharT __one=_CharT('1'), const allocator_type &__alloc=allocator_type())
- dynamic_bitset (dynamic_bitset &&__b) noexcept
- **dynamic_bitset** (initializer_list< block_type > __il, const allocator_type &__alloc=allocator_type())
- dynamic_bitset (size_type __nbits, unsigned long long __val=0ULL, const allocator_type &__alloc=allocator_type())
- template<typename _Traits = std::char_traits<char>, typename _CharT = typename _Traits::char_type>
  void **_M_copy_from_ptr** (const _CharT *, size_t, size_t, size_t, _CharT __zero=_CharT('0'), _CharT __one=_CharT('1'))
- template<typename _CharT, typename _Traits, typename _Alloc1>
  void **_M_copy_from_string** (const basic_string< _CharT, _Traits, _Alloc1 > &__str, size_t __pos, size_t __n, _CharT __zero=_CharT('0'), _CharT __one=_CharT('1'))

- template<typename _CharT, typename _Traits, typename _Alloc1>
  void **_M_copy_to_string** (std::basic_string< _CharT, _Traits, _Alloc1 > &__str, _CharT __zero=_CharT('0'), _CharT __one=_CharT('1')) const
- bool all () const
- bool any () const
- template<typename _BlockInputIterator>
  void append (_BlockInputIterator __first, _BlockInputIterator __last)
- void append (block_type __block)
- void **append** (initializer_list< block_type > __il)
- void clear ()
- size_type count () const noexcept
- bool empty () const noexcept
- size_type find_first () const
- size_type find_next (size_t __prev) const
- dynamic_bitset & flip ()
- dynamic_bitset & flip (size_type __pos)
- allocator_type get_allocator () const noexcept
- bool **is_proper_subset_of** (const dynamic_bitset &__b) const
- bool **is_subset_of** (const dynamic_bitset &__b) const
- constexpr size_type max_size () noexcept
- bool none () const
- size_type num_blocks () const noexcept
- dynamic_bitset & operator= (const dynamic_bitset &)=default
- dynamic_bitset & operator= (dynamic_bitset &&__b) noexcept(std::is_nothrow_move_assignable< _Base >::value)
- dynamic_bitset operator~ () const
- void push_back (bool __bit)
- dynamic_bitset & reset ()
- dynamic_bitset & reset (size_type __pos)
- void resize (size_type __nbits, bool __value=false)
- dynamic_bitset & set ()
- dynamic_bitset & set (size_type __pos, bool __val=true)
- size_type size () const noexcept
- void swap (dynamic_bitset &__b) noexcept
- bool test (size_type __pos) const
- template<typename _CharT = char, typename _Traits = std::char_traits<_CharT>, typename _Alloc1 = std::allocator<_CharT>>
  std::basic_string< _CharT, _Traits, _Alloc1 > to_string (_CharT __zero=_CharT('0'), _CharT __one=_CharT('1')) const
- unsigned long long to_ullong () const
- unsigned long to_ulong () const

- dynamic_bitset & operator&= (const dynamic_bitset &__rhs)
- dynamic_bitset & operator&= (dynamic_bitset &&__rhs)
- dynamic_bitset & operator|= (const dynamic_bitset &__rhs)
- dynamic_bitset & operator^= (const dynamic_bitset &__rhs)
- dynamic_bitset & operator-= (const dynamic_bitset &__rhs)

- dynamic_bitset & operator<<= (size_type __pos)
- dynamic_bitset & operator>>= (size_type __pos)

- reference operator[ ] (size_type __pos)
- const_reference operator[ ] (size_type __pos) const

- dynamic_bitset operator<< (size_type __pos) const
- dynamic_bitset operator>> (size_type __pos) const

Static Public Attributes

- static const size_type **bits_per_block**
- static const size_type **npos**

Friends

- bool **operator**< (const dynamic_bitset &__lhs, const dynamic_bitset &__rhs) noexcept
- bool **operator==** (const dynamic_bitset &__lhs, const dynamic_bitset &__rhs) noexcept
- class **reference**

5.396.1 Detailed Description

template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
class std::tr2::dynamic_bitset< _WordT, _Alloc >

The dynamic_bitset class represents a sequence of bits.
See N2050, Proposal to Add a Dynamically Sizeable Bitset to the Standard Library. http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2006/n2050.pdf
In the general unoptimized case, storage is allocated in word-sized blocks. Let B be the number of bits in a word, then (Nb+(B-1))/B words will be used for storage. B - NbB bits are unused. (They are the high-order bits in the highest word.) It is a class invariant that those unused bits are always zero.
If you think of dynamic_bitset as "a simple array of bits," be aware that your mental picture is reversed: a dynamic_bitset behaves the same way as bits in integers do, with the bit at index 0 in the "least significant / right-hand" position, and the bit at index Nb-1 in the "most significant / left-hand" position. Thus, unlike other containers, a dynamic_bitset's index "counts from right to left," to put it very loosely.
This behavior is preserved when translating to and from strings. For example, the first line of the following program probably prints "b('a') is 0001100001" on a modern ASCII system.

```
#include <dynamic_bitset>
#include <iostream>
#include <sstream>

using namespace std;

int main()
{
    long         a = 'a';
    dynamic_bitset<> b(a);

    cout « "b('a') is " « b « endl;

    ostringstream s;
    s « b;
    string  str = s.str();
    cout « "index 3 in the string is " « str[3] « " but\n"
         « "index 3 in the bitset is " « b[3] « endl;
}
```

Most of the actual code isn't contained in dynamic_bitset<> itself, but in the base class __dynamic_bitset_base. The base class works with whole words, not with individual bits. This allows us to specialize __dynamic_bitset_base for the important special case where the dynamic_bitset is only a single word.
Extra confusion can result due to the fact that the storage for __dynamic_bitset_base *is* a vector, and is indexed as such. This is carefully encapsulated.

5.396.2 Constructor & Destructor Documentation

dynamic_bitset() [1/7]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset ()  [default]
```

All bits set to zero.

dynamic_bitset() [2/7]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (
            const allocator_type & __alloc)  [inline], [explicit]
```

All bits set to zero.

dynamic_bitset() [3/7]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (
            size_type __nbits,
            unsigned long long __val = 0ULL,
            const allocator_type & __alloc = allocator_type())  [inline], [explicit]
```

Initial bits bitwise-copied from a single word (others set to zero).

dynamic_bitset() [4/7]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
template<typename _CharT, typename _Traits, typename _Alloc1>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (
            const std::basic_string< _CharT, _Traits, _Alloc1 > & __str,
            typename basic_string< _CharT, _Traits, _Alloc1 >::size_type __pos = 0,
            typename basic_string< _CharT, _Traits, _Alloc1 >::size_type __n = std::basic_string<↩
_CharT, _Traits, _Alloc1>::npos,
            _CharT __zero = _CharT('0'),
            _CharT __one = _CharT('1'),
            const allocator_type & __alloc = allocator_type())  [inline], [explicit]
```

Use a subset of a string.

Parameters

__str	A string of '0' and '1' characters.
__pos	Index of the first character in __str to use.
__n	The number of characters to copy.
__zero	The character to use for unset bits.
__one	The character to use for set bits.
__alloc	An allocator.

Exceptions

std::out_of_range	If __pos is bigger the size of __str.
std::invalid_argument	If a character appears in the string which is neither '0' nor '1'.

dynamic_bitset() [5/7]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (
            const char * __str,
            const allocator_type & __alloc = allocator_type())  [inline], [explicit]
```

Construct from a string.

Parameters

__str	A string of '0' and '1' characters.
__alloc	An allocator.

Exceptions

std::invalid_argument	If a character appears in the string which is neither '0' nor '1'.

dynamic_bitset() [6/7]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (
            const dynamic_bitset< _WordT, _Alloc > & )  [default]
```

Copy constructor.

dynamic_bitset() [7/7]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (
            dynamic_bitset< _WordT, _Alloc > && __b)  [inline], [noexcept]
```

Move constructor.

5.396.3 Member Function Documentation

all()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
bool std::tr2::dynamic_bitset< _WordT, _Alloc >::all () const  [inline]
```

Tests whether all the bits are on.

Returns

True if all the bits are set.

any()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
bool std::tr2::dynamic_bitset< _WordT, _Alloc >::any () const  [inline]
```

Tests whether any of the bits are on.

Returns

True if at least one bit is set.

append() [1/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
template<typename _BlockInputIterator>
void std::tr2::dynamic_bitset< _WordT, _Alloc >::append (
            _BlockInputIterator __first,
            _BlockInputIterator __last)  [inline]
```

Append an iterator range of blocks.

append() [2/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
void std::tr2::dynamic_bitset< _WordT, _Alloc >::append (
            block_type __block)  [inline]
```

Append a block.

clear()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
void std::tr2::dynamic_bitset< _WordT, _Alloc >::clear ()  [inline]
```

Clear the bitset.

count()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
size_type std::tr2::dynamic_bitset< _WordT, _Alloc >::count () const  [inline], [noexcept]
```

Returns the number of bits which are set.

empty()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
bool std::tr2::dynamic_bitset< _WordT, _Alloc >::empty () const  [inline], [nodiscard], [noexcept]
```

Returns true if the dynamic_bitset is empty.

find_first()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
size_type std::tr2::dynamic_bitset< _WordT, _Alloc >::find_first () const  [inline]
```

Finds the index of the first "on" bit.

Returns

The index of the first bit set, or size() if not found.

See also

find_next

find_next()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
size_type std::tr2::dynamic_bitset< _WordT, _Alloc >::find_next (
            size_t __prev) const  [inline]
```

Finds the index of the next "on" bit after prev.

Returns

The index of the next bit set, or size() if not found.

Parameters

__prev	Where to start searching.

See also

find_first

flip() [1/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::flip ()  [inline]
```

Toggles every bit to its opposite value.

flip() [2/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::flip (
            size_type __pos)  [inline]
```

Toggles a given bit to its opposite value.

Parameters

__pos	The index of the bit.

Exceptions

std::out_of_range	If *__pos* is bigger the size of the set.

get_allocator()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
allocator_type std::tr2::dynamic_bitset< _WordT, _Alloc >::get_allocator () const  [inline],
[noexcept]
```

Return the allocator for the bitset.

max_size()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
size_type std::tr2::dynamic_bitset< _WordT, _Alloc >::max_size ()  [inline], [constexpr], [noexcept]
```

Returns the maximum size of a dynamic_bitset object having the same type as ∗this. The real answer is max() ∗ bits_per_block but is likely to overflow.

none()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
bool std::tr2::dynamic_bitset< _WordT, _Alloc >::none () const  [inline]
```

Tests whether any of the bits are on.

Returns

True if none of the bits are set.

num_blocks()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
size_type std::tr2::dynamic_bitset< _WordT, _Alloc >::num_blocks () const  [inline], [noexcept]
```

Returns the total number of blocks.

operator&=() [1/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::operator&= (
            const dynamic_bitset< _WordT, _Alloc > & __rhs)  [inline]
```

Operations on dynamic_bitsets.

Parameters

__rhs	A same-sized dynamic_bitset.

These should be self-explanatory.

operator&=() [2/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::operator&= (
            dynamic_bitset< _WordT, _Alloc > && __rhs)  [inline]
```

Operations on dynamic_bitsets.

Parameters

__rhs	A same-sized dynamic_bitset.

These should be self-explanatory.

operator-=()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::operator-= (
            const dynamic_bitset< _WordT, _Alloc > & __rhs)  [inline]
```

Operations on dynamic_bitsets.

Parameters

__rhs	A same-sized dynamic_bitset.

These should be self-explanatory.

operator<<()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset std::tr2::dynamic_bitset< _WordT, _Alloc >::operator<< (
            size_type __pos) const  [inline]
```

Self-explanatory.

operator<<=()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::operator<<= (
            size_type __pos)  [inline]
```

Operations on dynamic_bitsets.

Parameters

__pos	The number of places to shift.

These should be self-explanatory.

operator=() [1/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::operator= (
            const dynamic_bitset< _WordT, _Alloc > & )  [default]
```

Copy assignment operator.

operator=() [2/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::operator= (
            dynamic_bitset< _WordT, _Alloc > && __b)  [inline], [noexcept]
```

Move assignment operator.

operator>>()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset std::tr2::dynamic_bitset< _WordT, _Alloc >::operator>> (
            size_type __pos) const  [inline]
```

Self-explanatory.

operator>>=()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::operator>>= (
            size_type __pos)  [inline]
```

Operations on dynamic_bitsets.

Parameters

__pos	The number of places to shift.

These should be self-explanatory.

operator[]() [1/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
reference std::tr2::dynamic_bitset< _WordT, _Alloc >::operator[] (
            size_type __pos)  [inline]
```

Array-indexing support.

Parameters

__pos	Index into the dynamic_bitset.

Returns

A bool for a 'const dynamic_bitset'. For non-const bitsets, an instance of the reference proxy class.

Note

These operators do no range checking and throw no exceptions, as required by DR 11 to the standard.

operator[]() [2/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
const_reference std::tr2::dynamic_bitset< _WordT, _Alloc >::operator[] (
            size_type __pos) const  [inline]
```

Array-indexing support.

Parameters

__pos	Index into the dynamic_bitset.

Returns

A bool for a 'const dynamic_bitset'. For non-const bitsets, an instance of the reference proxy class.

Note

These operators do no range checking and throw no exceptions, as required by DR 11 to the standard.

operator^=()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::operator^= (
            const dynamic_bitset< _WordT, _Alloc > & __rhs)  [inline]
```

Operations on dynamic_bitsets.

Parameters

__rhs	A same-sized dynamic_bitset.

These should be self-explanatory.

operator"" |=()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::operator|= (
            const dynamic_bitset< _WordT, _Alloc > & __rhs)  [inline]
```

Operations on dynamic_bitsets.

Parameters

__rhs	A same-sized dynamic_bitset.

These should be self-explanatory.

operator~()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset std::tr2::dynamic_bitset< _WordT, _Alloc >::operator~ () const  [inline]
```

See the no-argument flip().

push_back()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
void std::tr2::dynamic_bitset< _WordT, _Alloc >::push_back (
            bool __bit)  [inline]
```

Push a bit onto the high end of the bitset.

reset() [1/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::reset ()  [inline]
```

Sets every bit to false.

reset() [2/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::reset (
            size_type __pos)  [inline]
```

Sets a given bit to false.

Parameters

__pos	The index of the bit.

Exceptions

std::out_of_range	If *__pos* is bigger the size of the set.

Same as writing `set(__pos, false)`.

resize()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
void std::tr2::dynamic_bitset< _WordT, _Alloc >::resize (
            size_type __nbits,
            bool __value = false)  [inline]
```

Resize the bitset.

Referenced by std::tr2::operator>>().

set() [1/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::set ()  [inline]
```

Sets every bit to true.

set() [2/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::set (
            size_type __pos,
            bool __val = true)  [inline]
```

Sets a given bit to a particular value.

Parameters

__pos	The index of the bit.
__val	Either true or false, defaults to true.

Exceptions

std::out_of_range	If *__pos* is bigger the size of the set.

size()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
size_type std::tr2::dynamic_bitset< _WordT, _Alloc >::size () const  [inline], [noexcept]
```

Returns the total number of bits.

Referenced by std::tr2::operator>>().

swap()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
void std::tr2::dynamic_bitset< _WordT, _Alloc >::swap (
            dynamic_bitset< _WordT, _Alloc > & __b)  [inline], [noexcept]
```

Swap with another bitset.

test()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
bool std::tr2::dynamic_bitset< _WordT, _Alloc >::test (
            size_type __pos) const  [inline]
```

Tests the value of a bit.

Parameters

__pos	The index of a bit.

Returns

The value at *__pos*.

Exceptions

std::out_of_range	If *__pos* is bigger the size of the set.

to_string()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
template<typename _CharT = char, typename _Traits = std::char_traits<_CharT>, typename _Alloc1 =
std::allocator<_CharT>>
std::basic_string< _CharT, _Traits, _Alloc1 > std::tr2::dynamic_bitset< _WordT, _Alloc >::to_↩
string (
            _CharT __zero = _CharT('0'),
            _CharT __one = _CharT('1')) const  [inline]
```

Returns a character interpretation of the dynamic_bitset.

Returns

The string equivalent of the bits.

Note the ordering of the bits: decreasing character positions correspond to increasing bit positions (see the main class notes for an example).

to_ullong()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
unsigned long long std::tr2::dynamic_bitset< _WordT, _Alloc >::to_ullong () const  [inline]
```

Returns a numerical interpretation of the dynamic_bitset.

Returns

The integral equivalent of the bits.

Exceptions

std::overflow_error	If there are too many bits to be represented in an `unsigned long`.

to_ulong()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
unsigned long std::tr2::dynamic_bitset< _WordT, _Alloc >::to_ulong () const  [inline]
```

Returns a numerical interpretation of the dynamic_bitset.

Returns

The integral equivalent of the bits.

Exceptions

std::overflow_error	If there are too many bits to be represented in an `unsigned long`.

The documentation for this class was generated from the following files:

- dynamic_bitset
- dynamic_bitset.tcc

5.397 std::enable_if< bool, _Tp > Struct Template Reference

`#include <type_traits>`

5.397.1 Detailed Description

template<bool, typename _Tp = void>
struct std::enable_if< bool, _Tp >

Define a member typedef `type` only if a boolean constant is true.
The documentation for this struct was generated from the following file:

- type_traits

5.398 std::enable_shared_from_this< _Tp > Class Template Reference

`#include <memory>`

Public Member Functions

- shared_ptr< _Tp > **shared_from_this** ()
- shared_ptr< const _Tp > **shared_from_this** () const

Protected Member Functions

- **enable_shared_from_this** (const enable_shared_from_this &) noexcept
- enable_shared_from_this & **operator=** (const enable_shared_from_this &) noexcept

Friends

- const enable_shared_from_this * **__enable_shared_from_this_base** (const __shared_count<> &, const enable_shared_from_this *__p)
- template<typename, _Lock_policy>
  class **__shared_ptr**

5.398.1 Detailed Description

template<typename _Tp>
class std::enable_shared_from_this< _Tp >

Base class allowing use of the member function `shared_from_this`.

Since

C++11

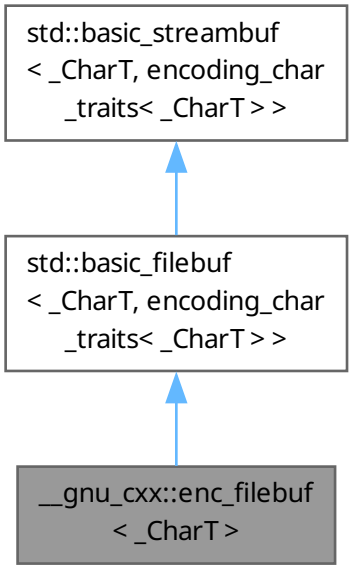
The documentation for this class was generated from the following file:

- bits/shared_ptr.h

5.399 __gnu_cxx::enc_filebuf< _CharT > Class Template Reference

`#include <enc_filebuf.h>`

Inheritance diagram for __gnu_cxx::enc_filebuf< _CharT >:



Public Types

- typedef codecvt< char_type, char, __state_type > **__codecvt_type**
- typedef __basic_file< char > **__file_type**
- typedef basic_filebuf< char_type, traits_type > **__filebuf_type**
- typedef traits_type::state_type **__state_type**
- typedef basic_streambuf< char_type, traits_type > **__streambuf_type**
- typedef _CharT **char_type**
- typedef traits_type::int_type **int_type**
- typedef traits_type::off_type **off_type**
- typedef traits_type::pos_type **pos_type**
- typedef traits_type::state_type **state_type**
- typedef encoding_char_traits< _CharT > **traits_type**

Public Member Functions

- **enc_filebuf** (state_type &__state)
- __filebuf_type * close ()
- locale getloc () const
- streamsize in_avail ()
- bool is_open () const throw ()
- _If_fs_path< _Path, __filebuf_type * > open (const _Path &__s, ios_base::openmode __mode)
- __filebuf_type * open (const char *__s, ios_base::openmode __mode)
- __filebuf_type * open (const std::string &__s, ios_base::openmode __mode)
- locale pubimbue (const locale &__loc)

- int_type sbumpc ()
- int_type sgetc ()
- streamsize sgetn (char_type *__s, streamsize __n)
- int_type snextc ()
- int_type sputbackc (char_type __c)
- int_type sputc (char_type __c)
- streamsize sputn (const char_type *__s, streamsize __n)
- int_type sungetc ()
- void **swap** (basic_filebuf &)

- basic_streambuf * pubsetbuf (char_type *__s, streamsize __n)
- pos_type pubseekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)
- pos_type pubseekpos (pos_type __sp, ios_base::openmode __mode=ios_base::in|ios_base::out)
- int pubsync ()

Protected Member Functions

- void **__safe_gbump** (streamsize __n)
- void **__safe_pbump** (streamsize __n)
- void **_M_allocate_internal_buffer** ()
- bool **_M_convert_to_external** (char_type *, streamsize)
- void _M_create_pback ()
- void **_M_destroy_internal_buffer** () throw ()
- void _M_destroy_pback () throw ()
- int **_M_get_ext_pos** (__state_type &__state)
- pos_type **_M_seek** (off_type __off, ios_base::seekdir __way, __state_type __state)
- void _M_set_buffer (streamsize __off)
- bool **_M_terminate_output** ()
- void gbump (int __n)
- virtual void imbue (const locale &__loc)
- virtual int_type overflow (int_type __c=encoding_char_traits< _CharT >::eof())
- virtual int_type pbackfail (int_type __c=encoding_char_traits< _CharT >::eof())
- void pbump (int __n)
- virtual pos_type seekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)
- virtual pos_type seekpos (pos_type __pos, ios_base::openmode __mode=ios_base::in|ios_base::out)
- virtual __streambuf_type * setbuf (char_type *__s, streamsize __n)
- void setg (char_type *__gbeg, char_type *__gnext, char_type *__gend)
- void setp (char_type *__pbeg, char_type *__pend)
- virtual streamsize showmanyc ()
- void **swap** (basic_streambuf &__sb)
- virtual int sync ()
- virtual int_type uflow ()
- virtual int_type underflow ()
- virtual streamsize xsgetn (char_type *__s, streamsize __n)
- virtual streamsize xsputn (const char_type *__s, streamsize __n)

- char_type * eback () const
- char_type * gptr () const

- char_type * egptr () const

- char_type * pbase () const
- char_type * pptr () const
- char_type * epptr () const

Protected Attributes

- char_type * _M_buf
- bool **_M_buf_allocated**
- locale _M_buf_locale
- size_t _M_buf_size
- const __codecvt_type * **_M_codecvt**
- char * _M_ext_buf
- streamsize _M_ext_buf_size
- char * **_M_ext_end**
- const char * _M_ext_next
- __file_type **_M_file**
- char_type * _M_in_beg
- char_type * _M_in_cur
- char_type * _M_in_end
- __c_lock **_M_lock**
- ios_base::openmode _M_mode
- char_type * _M_out_beg
- char_type * _M_out_cur
- char_type * _M_out_end
- bool _M_reading
- __state_type **_M_state_beg**
- __state_type **_M_state_cur**
- __state_type **_M_state_last**
- bool **_M_writing**

- char_type _M_pback
- char_type * _M_pback_cur_save
- char_type * _M_pback_end_save
- bool _M_pback_init

5.399.1 Detailed Description

template<typename _CharT>
class __gnu_cxx::enc_filebuf< _CharT >

class enc_filebuf.

5.399.2 Member Function Documentation

_M_create_pback()

`void std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_create_pback () [inline], [protected], [inherited]`

Initializes pback buffers, and moves normal buffers to safety. Assumptions: _M_in_cur has already been moved back

_M_destroy_pback()

```
void std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_destroy_pback () throw ( )
[inline], [protected], [inherited]
```

Deactivates pback buffer contents, and restores normal buffer. Assumptions: The pback buffer has only moved forward.

_M_set_buffer()

```
void std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_set_buffer (
            streamsize __off)  [inline], [protected], [inherited]
```

This function sets the pointers of the internal buffer, both get and put areas. Typically:
__off == egptr() - eback() upon underflow/uflow (**read** mode); __off == 0 upon overflow (**write** mode); __off == -1 upon open, setbuf, seekoff/pos (**uncommitted** mode).
NB: epptr() - pbase() == _M_buf_size - 1, since _M_buf_size reflects the actual allocated memory and the last cell is reserved for the overflow char of a full put area.

close()

```
basic_filebuf< _CharT, encoding_char_traits< _CharT > >::__filebuf_type * std::basic_filebuf< ↩
_CharT, encoding_char_traits< _CharT > >::close ()  [inherited]
```

Closes the currently associated file.

Returns

`this` on success, NULL on failure

If no file is currently open, this function immediately fails.
If a *put buffer area* exists, `overflow(eof)` is called to flush all the characters. The file is then closed.
If any operations fail, this function also fails.

eback()

```
char_type * std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::eback () const  [inline],
[protected], [inherited]
```

Access to the get area.
These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

egptr()

```
char_type * std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::egptr () const  [inline],
[protected], [inherited]
```

Access to the get area.
These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

epptr()

```
char_type * std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::epptr () const  [inline],
[protected], [inherited]
```

Access to the put area.
These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

gbump()

```
void std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::gbump (
            int __n)  [inline], [protected], [inherited]
```

Moving the read position.

Parameters

_↩ _n	The delta by which to move.

This just advances the read position without returning any data.

getloc()

```
locale std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::getloc () const  [inline],
[inherited]
```

Locale access.

Returns

The current locale in effect.

If pubimbue(loc) has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

gptr()

```
char_type * std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::gptr () const  [inline],
[protected], [inherited]
```

Access to the get area.
These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

imbue()

```
void std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::imbue (
            const locale & __loc)  [protected], [virtual], [inherited]
```

Changes translations.

Parameters

__loc	A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

Note

Base class version does nothing.

Reimplemented from std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >.

in_avail()

```
streamsize std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::in_avail ()  [inline], [inherited]
```

Looking ahead into the stream.

Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

is_open()

```
bool std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::is_open () const throw ( ) [inline], [nodiscard], [inherited]
```

Returns true if the external file is open.

open() [1/3]

```
_If_fs_path< _Path, __filebuf_type * > std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::open (
            const _Path & __s,
            ios_base::openmode __mode)  [inline], [inherited]
```

Opens an external file.

Parameters

__s	The name of the file, as a filesystem::path.
__mode	The open mode flags.

Returns

`this` on success, NULL on failure

open() [2/3]

```
basic_filebuf< _CharT, encoding_char_traits< _CharT > >::__filebuf_type * std::basic_filebuf< ↩
_CharT, encoding_char_traits< _CharT > >::open (
            const char * __s,
            ios_base::openmode __mode)  [inherited]
```

Opens an external file.

Parameters

__s	The name of the file.
__mode	The open mode flags.

Returns

`this` on success, NULL on failure

If a file is already open, this function immediately fails. Otherwise it tries to open the file named *__s* using the flags given in *__mode*.
Table 92, adapted here, gives the relation between openmode combinations and the equivalent `fopen()` flags. (NB: lines app, in|out|app, in|app, binary|app, binary|in|out|app, and binary|in|app per DR 596)

ios_base Flag combination				stdio equivalent
binary	in	out	trunc	app	
	+			w
	+		+	a
			+	a
	+	+		w
+				r
+	+			r+
+	+	+		w+
+	+		+	a+
+			+	a+
+	+			wb
+	+		+	ab
+			+	ab
+	+	+		wb
+	+			rb
+	+	+		r+b
+	+	+	+	w+b
+	+	+	+	a+b
+	+		+	a+b

open() [3/3]

```
__filebuf_type * std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::open (
            const std::string & __s,
            ios_base::openmode __mode)  [inline], [inherited]
```

Opens an external file.

Parameters

__s	The name of the file.
__mode	The open mode flags.

Returns

`this` on success, NULL on failure

overflow()

```
basic_filebuf< _CharT, encoding_char_traits< _CharT > >::int_type std::basic_filebuf< _CharT,
encoding_char_traits< _CharT > >::overflow (
            int_type __c = _Traits::eof())  [protected], [virtual], [inherited]
```

Consumes data from the buffer; writes to the controlled sequence.

Parameters

_↩ _c	An additional character to consume.

Returns

eof() to indicate failure, something else (usually __c, or not_eof())

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if __c is not `eof()`.
For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.
A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >.

pbackfail()

```
basic_filebuf< _CharT, encoding_char_traits< _CharT > >::int_type std::basic_filebuf< _CharT,
encoding_char_traits< _CharT > >::pbackfail (
            int_type __c = _Traits::eof())  [protected], [virtual], [inherited]
```

Tries to back up the input sequence.

Parameters

_↩ _c	The character to be inserted back into the sequence.

Returns

eof() on failure, *some other value* on success

Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >.

pbase()

```
char_type * std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::pbase () const  [inline],
[protected], [inherited]
```

Access to the put area.
These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

pbump()

```
void std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::pbump (
            int __n)  [inline], [protected], [inherited]
```

Moving the write position.

Parameters

_↩ _n	The delta by which to move.

This just advances the write position without returning any data.

pptr()

```
char_type * std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::pptr () const  [inline],
[protected], [inherited]
```

Access to the put area.
These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

pubimbue()

```
locale std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::pubimbue (
            const locale & __loc)  [inline], [inherited]
```

Entry point for imbue().

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls the derived imbue(__loc).

pubseekoff()

```
pos_type std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::pubseekoff (
            off_type __off,
            ios_base::seekdir __way,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [inline], [inherited]
```

Alters the stream position.

Parameters

__off	Offset.
__way	Value for ios_base::seekdir.
__mode	Value for ios_base::openmode.

Calls virtual seekoff function.

pubseekpos()

```
pos_type std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::pubseekpos (
            pos_type __sp,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [inline], [inherited]
```

Alters the stream position.

Parameters

__sp	Position
__mode	Value for ios_base::openmode.

Calls virtual seekpos function.

pubsetbuf()

```
basic_streambuf * std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::pubsetbuf (
            char_type * __s,
            streamsize __n)  [inline], [inherited]
```

Entry points for derived buffer functions.
The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

pubsync()

```
int std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::pubsync ()  [inline], [inherited]
```

Calls virtual sync function.

sbumpc()

```
int_type std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::sbumpc ()  [inline],
[inherited]
```

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

seekoff()

```
basic_filebuf< _CharT, encoding_char_traits< _CharT > >::pos_type std::basic_filebuf< _CharT,
encoding_char_traits< _CharT > >::seekoff (
            off_type __off,
            ios_base::seekdir __way,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [protected], [virtual],
[inherited]
```

Alters the stream positions.
Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >.

seekpos()

```
basic_filebuf< _CharT, encoding_char_traits< _CharT > >::pos_type std::basic_filebuf< _CharT,
encoding_char_traits< _CharT > >::seekpos (
            pos_type __pos,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [protected], [virtual],
[inherited]
```

Alters the stream positions.
Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >.

setbuf()

```
basic_filebuf< _CharT, encoding_char_traits< _CharT > >::__streambuf_type * std::basic_filebuf<
_CharT, encoding_char_traits< _CharT > >::setbuf (
            char_type * __s,
            streamsize __n)  [protected], [virtual], [inherited]
```

Manipulates the buffer.

Parameters

__s	Pointer to a buffer area.
__n	Size of *__s*.

Returns

`this`

If no file has been opened, and both *__s* and *__n* are zero, then the stream becomes unbuffered. Otherwise, `__s` is used as a buffer; see https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html#io.streambuf.buffering for more.
Reimplemented from std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >.

setg()

```
void std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::setg (
            char_type * __gbeg,
            char_type * __gnext,
            char_type * __gend)  [inline], [protected], [inherited]
```

Setting the three read area pointers.

Parameters

__gbeg	A pointer.
__gnext	A pointer.
__gend	A pointer.

Postcondition

__gbeg == `eback()`, *__gnext* == `gptr()`, and *__gend* == `egptr()`

setp()

```
void std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::setp (
            char_type * __pbeg,
            char_type * __pend)  [inline], [protected], [inherited]
```

Setting the three write area pointers.

Parameters

__pbeg	A pointer.
__pend	A pointer.

Postcondition

__pbeg == `pbase()`, *__pbeg* == `pptr()`, and *__pend* == `epptr()`

sgetc()

```
int_type std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::sgetc ()  [inline],
[inherited]
```

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

sgetn()

```
streamsize std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::sgetn (
            char_type * __s,
            streamsize __n)  [inline], [inherited]
```

Entry point for xsgetn.

Parameters

_↩ _s	A buffer area.
_↩ _n	A count.

Returns xsgetn(__s,__n). The effect is to fill __s[0] through __s[__n-1] with characters from the input sequence, if possible.

showmanyc()

`streamsize std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::showmanyc () [protected], [virtual], [inherited]`

Investigating the data available.

Returns

An estimate of the number of characters available in the input sequence, or -1.

If it returns a positive value, then successive calls to `underflow()` *will not return* `traits::eof()` *until at least that number of characters have been supplied. If* `showmanyc()` *returns -1, then calls to* `underflow()` *or* `uflow()` *will fail.* [27.5.2.4.3]/1

Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to underflow or uflow] will not return* `eof()` *but that they will return immediately.*

The standard adds that *the morphemes of* `showmanyc` *are* ***es-how-many-see****, not* ***show-manic****.*

Reimplemented from std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >.

snextc()

`int_type std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::snextc () [inline], [inherited]`

Getting the next character.

Returns

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

sputbackc()

```
int_type std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::sputbackc (
            char_type __c) [inline], [inherited]
```

Pushing characters back into the input stream.

Parameters

_↩ _c	The character to push back.

Returns

The previous character, if possible.

Similar to sungetc(), but __*c* is pushed onto the stream instead of *the previous character.* If successful, the next character fetched from the input stream will be __*c*.

sputc()

```
int_type std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::sputc (
            char_type __c)  [inline], [inherited]
```

Entry point for all single-character output functions.

Parameters

_↩ _*c*	A character to output.

Returns

__*c*, if possible.

One of two public output functions.
If a write position is available for the output sequence (i.e., the buffer is not full), stores __*c* in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(↩ __c)`.

sputn()

```
streamsize std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::sputn (
            const char_type * __s,
            streamsize __n)  [inline], [inherited]
```

Entry point for all single-character output functions.

Parameters

_↩ _*s*	A buffer read area.
_↩ _*n*	A count.

One of two public output functions.
Returns xsputn(__s,__n). The effect is to write __*s*[0] through __*s*[__n-1] to the output sequence, if possible.

sungetc()

```
int_type std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::sungetc ()  [inline],
[inherited]
```

Moving backwards in the input stream.

Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns pbackfail(). The effect is to *unget* the last character *gotten*.

sync()

```
int std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::sync ()  [protected], [virtual],
[inherited]
```

Synchronizes the buffer arrays with the controlled sequences.

Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

Note

Base class version does nothing, returns zero.

Reimplemented from std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >.

uflow()

```
virtual int_type std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::uflow ()  [inline],
[protected], [virtual], [inherited]
```

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

underflow()

```
basic_filebuf< _CharT, encoding_char_traits< _CharT > >::int_type std::basic_filebuf< _CharT,
encoding_char_traits< _CharT > >::underflow ()  [protected], [virtual], [inherited]
```

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.
For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.
A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >.

xsgetn()

```
streamsize std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::xsgetn (
            char_type * __s,
            streamsize __n)  [protected], [virtual], [inherited]
```

Multiple character extraction.

Parameters

_↩ _s	A buffer area.
_↩ _n	Maximum number of characters to assign.

Returns

The number of characters assigned.

Fills __s[0] through __s[__n-1] with characters from the input sequence, as if by `sbumpc()`. Stops when either __n characters have been copied, or when `traits::eof()` would be copied.
It is expected that derived classes provide a more efficient implementation by overriding this definition.
Reimplemented from std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >.

xsputn()

```
streamsize std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::xsputn (
            const char_type * __s,
            streamsize __n)  [protected], [virtual], [inherited]
```

Multiple character insertion.

Parameters

_↩ _s	A buffer area.
_↩ _n	Maximum number of characters to write.

Returns

The number of characters written.

Writes __s[0] through __s[__n-1] to the output sequence, as if by `sputc()`. Stops when either *n* characters have been copied, or when `sputc()` would return `traits::eof()`.
It is expected that derived classes provide a more efficient implementation by overriding this definition.
Reimplemented from std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >.

5.399.3 Member Data Documentation

_M_buf

```
char_type* std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_buf  [protected],
[inherited]
```

Pointer to the beginning of internal buffer.

_M_buf_locale

```
locale std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::_M_buf_locale  [protected],
[inherited]
```

Current locale setting.

_M_buf_size

size_t std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_buf_size [protected], [inherited]

Actual size of internal buffer. This number is equal to the size of the put area + 1 position, reserved for the overflow char of a full area.

_M_ext_buf

char* std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_ext_buf [protected], [inherited]

Buffer for external characters. Used for input when codecvt::always_noconv() == false. When valid, this corresponds to eback().

_M_ext_buf_size

streamsize std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_ext_buf_size [protected], [inherited]

Size of buffer held by _M_ext_buf.

_M_ext_next

const char* std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_ext_next [protected], [inherited]

Pointers into the buffer held by _M_ext_buf that delimit a subsequence of bytes that have been read but not yet converted. When valid, _M_ext_next corresponds to egptr().

_M_in_beg

char_type* std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::_M_in_beg [protected], [inherited]

Start of get area.

_M_in_cur

char_type* std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::_M_in_cur [protected], [inherited]

Current read area.

_M_in_end

char_type* std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::_M_in_end [protected], [inherited]

End of get area.

_M_mode

ios_base::openmode std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_mode [protected], [inherited]

Place to stash in || out || in | out settings for current filebuf.

_M_out_beg

char_type* std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::_M_out_beg [protected], [inherited]

Start of put area.

_M_out_cur

char_type* std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::_M_out_cur [protected], [inherited]

Current put area.

_M_out_end

char_type* std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::_M_out_end [protected], [inherited]

End of put area.

_M_pback

char_type std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_pback [protected], [inherited]

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

_M_pback_cur_save

char_type* std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_pback_cur_save [protected], [inherited]

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

_M_pback_end_save

char_type* std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_pback_end_save [protected], [inherited]

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

_M_pback_init

bool std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_pback_init [protected], [inherited]

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

_M_reading

bool std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_reading [protected], [inherited]

_M_reading == false && _M_writing == false for **uncommitted** mode; _M_reading == true for **read** mode; _M_writing == true for **write** mode;

NB: _M_reading == true && _M_writing == true is unused.

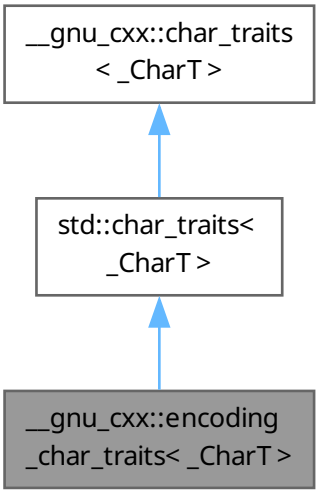
The documentation for this class was generated from the following file:

- enc_filebuf.h

5.400 __gnu_cxx::encoding_char_traits< _CharT > Struct Template Reference

#include <codecvt_specializations.h>

Inheritance diagram for __gnu_cxx::encoding_char_traits< _CharT >:



Public Types

- typedef _CharT **char_type**
- using **comparison_category**
- typedef _Char_types< _CharT >::int_type **int_type**
- typedef _Char_types< _CharT >::off_type **off_type**
- typedef std::fpos< state_type > **pos_type**
- typedef encoding_state **state_type**

Static Public Member Functions

- static constexpr void **assign** (char_type &__c1, const char_type &__c2)
- static constexpr char_type * **assign** (char_type *__s, std::size_t __n, char_type __a)
- static constexpr int **compare** (const char_type *__s1, const char_type *__s2, std::size_t __n)
- static constexpr char_type * **copy** (char_type *__s1, const char_type *__s2, std::size_t __n)

- static constexpr int_type **eof** ()
- static constexpr bool **eq** (const char_type &__c1, const char_type &__c2)
- static constexpr bool **eq_int_type** (const int_type &__c1, const int_type &__c2)
- static constexpr const char_type * **find** (const char_type *__s, std::size_t __n, const char_type &__a)
- static constexpr std::size_t **length** (const char_type *__s)
- static constexpr bool **lt** (const char_type &__c1, const char_type &__c2)
- static constexpr char_type * **move** (char_type *__s1, const char_type *__s2, std::size_t __n)
- static constexpr int_type **not_eof** (const int_type &__c)
- static constexpr char_type **to_char_type** (const int_type &__c)
- static constexpr int_type **to_int_type** (const char_type &__c)

5.400.1 Detailed Description

template<**typename _CharT**>
struct __gnu_cxx::encoding_char_traits< **_CharT** >

encoding_char_traits
The documentation for this struct was generated from the following file:

- codecvt_specializations.h

5.401 __gnu_cxx::encoding_state Class Reference

`#include <codecvt_specializations.h>`

Public Types

- typedef iconv_t **descriptor_type**

Public Member Functions

- **encoding_state** (const char *__int, const char *__ext, int __ibom=0, int __ebom=0, int __bytes=1)
- **encoding_state** (const encoding_state &__obj)
- int **character_ratio** () const
- int **external_bom** () const
- const std::string **external_encoding** () const
- bool **good** () const throw ()
- const descriptor_type & **in_descriptor** () const
- int **internal_bom** () const
- const std::string **internal_encoding** () const
- encoding_state & **operator=** (const encoding_state &__obj)
- const descriptor_type & **out_descriptor** () const

Protected Member Functions

- void **construct** (const encoding_state &__obj)
- void **destroy** () throw ()
- void **init** ()

Protected Attributes

- int **_M_bytes**
- int **_M_ext_bom**
- std::string **_M_ext_enc**
- descriptor_type **_M_in_desc**
- int **_M_int_bom**
- std::string **_M_int_enc**
- descriptor_type **_M_out_desc**

5.401.1 Detailed Description

Extension to use iconv for dealing with character encodings.
The documentation for this class was generated from the following file:

- codecvt_specializations.h

5.402 std::encoding_state Class Reference

`#include <codecvt_specializations.h>`

Public Types

- typedef iconv_t **descriptor_type**

Public Member Functions

- **encoding_state** (const char *__int, const char *__ext, int __ibom=0, int __ebom=0, int __bytes=1)
- **encoding_state** (const encoding_state &__obj)
- int **character_ratio** () const
- int **external_bom** () const
- const std::string **external_encoding** () const
- bool **good** () const throw ()
- const descriptor_type & **in_descriptor** () const
- int **internal_bom** () const
- const std::string **internal_encoding** () const
- encoding_state & **operator=** (const encoding_state &__obj)
- const descriptor_type & **out_descriptor** () const

Protected Member Functions

- void **construct** (const encoding_state &__obj)
- void **destroy** () throw ()
- void **init** ()

Protected Attributes

- int **_M_bytes**
- int **_M_ext_bom**
- std::string **_M_ext_enc**
- descriptor_type **_M_in_desc**
- int **_M_int_bom**
- std::string **_M_int_enc**
- descriptor_type **_M_out_desc**

5.402.1 Detailed Description

Extension to use iconv for dealing with character encodings.
The documentation for this class was generated from the following file:

- codecvt_specializations.h

5.403 __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, No_Throw > Struct Template Reference

5.403.1 Detailed Description

template<typename _VTp, typename Cmp_Fn, typename _Alloc, bool No_Throw>
struct __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, No_Throw >

Entry compare, primary template.
The documentation for this struct was generated from the following file:

- entry_cmp.hpp

5.404 __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, false > Struct Template Reference

`#include <entry_cmp.hpp>`

Classes

- struct type

Public Types

- typedef __rebind_v::const_pointer **entry**

5.404.1 Detailed Description

template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, false >

Specialization, false.
The documentation for this struct was generated from the following file:

- entry_cmp.hpp

5.405 __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, true > Struct Template Reference

`#include <entry_cmp.hpp>`

Public Types

- typedef Cmp_Fn type

5.405.1 Detailed Description

template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, true >

Specialization, true.

5.405.2 Member Typedef Documentation

type

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
typedef Cmp_Fn __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, true >::type
```

Compare.
The documentation for this struct was generated from the following file:

- entry_cmp.hpp

5.406 __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, No_Throw > Struct Template Reference

5.406.1 Detailed Description

template<typename _VTp, typename Pred, typename _Alloc, bool No_Throw>
struct __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, No_Throw >

Entry predicate primary class template.
The documentation for this struct was generated from the following file:

- entry_pred.hpp

5.407 __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, false > Struct Template Reference

```
#include <entry_pred.hpp>
```

Public Types

- typedef __rebind_v::const_pointer **entry**

5.407.1 Detailed Description

template<typename _VTp, typename Pred, typename _Alloc>
struct __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, false >

Specialization, false.
The documentation for this struct was generated from the following file:

- entry_pred.hpp

5.408 __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, true > Struct Template Reference

```
#include <entry_pred.hpp>
```

Public Types

- typedef Pred **type**

5.408.1 Detailed Description

template<typename _VTp, typename Pred, typename _Alloc>
struct __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, true >

Specialization, true.
The documentation for this struct was generated from the following file:

- entry_pred.hpp

5.409 __gnu_pbds::detail::eq_by_less< Key, Cmp_Fn > Struct Template Reference

`#include <eq_by_less.hpp>`

Public Member Functions

- bool **operator()** (const Key &r_lhs, const Key &r_rhs) const

5.409.1 Detailed Description

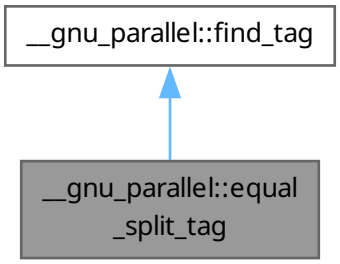
template<typename Key, class Cmp_Fn>
struct __gnu_pbds::detail::eq_by_less< Key, Cmp_Fn >

Equivalence function.
The documentation for this struct was generated from the following file:

- eq_by_less.hpp

5.410 __gnu_parallel::equal_split_tag Struct Reference

`#include <tags.h>`
Inheritance diagram for __gnu_parallel::equal_split_tag:



5.410.1 Detailed Description

Selects the equal splitting variant for std::find().

See also

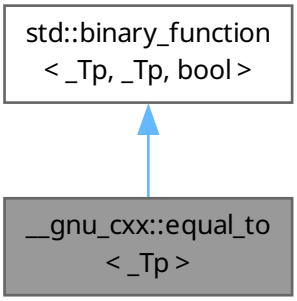
_GLIBCXX_FIND_EQUAL_SPLIT

The documentation for this struct was generated from the following file:

- tags.h

5.411 __gnu_cxx::equal_to< _Tp > Struct Template Reference

`#include <stl_function.h>`
Inheritance diagram for __gnu_cxx::equal_to< _Tp >:



Public Types

- typedef _Tp first_argument_type
- typedef bool result_type
- typedef _Tp second_argument_type

Public Member Functions

- constexpr bool **operator()** (const _Tp &__x, const _Tp &__y) const

5.411.1 Detailed Description

template<typename _Tp>
struct __gnu_cxx::equal_to< _Tp >

One of the comparison functors.

5.411.2 Member Typedef Documentation

first_argument_type

`typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type [inherited]`
`first_argument_type` is the type of the first argument

result_type

`typedef bool std::binary_function< _Tp, _Tp, bool >::result_type [inherited]`
`result_type` is the return type

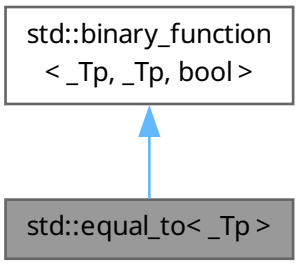
second_argument_type

`typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type [inherited]`
`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

5.412 std::equal_to< _Tp > Struct Template Reference

`#include <stl_function.h>`
Inheritance diagram for std::equal_to< _Tp >:



Public Types

- typedef _Tp first_argument_type
- typedef bool result_type
- typedef _Tp second_argument_type

Public Member Functions

- constexpr bool **operator()** (const _Tp &__x, const _Tp &__y) const

5.412.1 Detailed Description

template<typename _Tp>
struct std::equal_to< _Tp >

One of the comparison functors.

5.412.2 Member Typedef Documentation

first_argument_type

`typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type [inherited]`
`first_argument_type` is the type of the first argument

result_type

`typedef bool std::binary_function< _Tp, _Tp, bool >::result_type [inherited]`
`result_type` is the return type

second_argument_type

`typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type [inherited]`
`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

5.413 std::ranges::equal_to Struct Reference

`#include <ranges_cmp.h>`

Public Types

- using **is_transparent**

Public Member Functions

- template<typename _Tp, typename _Up>
  requires equality_comparable_with<_Tp, _Up>
  constexpr bool **operator()** (_Tp &&__t, _Up &&__u) const noexcept(noexcept(std::declval< _Tp >()==std::declval< _Up >()))

5.413.1 Detailed Description

ranges::equal_to function object type.
The documentation for this struct was generated from the following file:

- ranges_cmp.h

5.414 std::error_category Class Reference

`#include <system_error>`

Public Member Functions

- **error_category** (const error_category &)=delete
- virtual error_condition default_error_condition (int __i) const noexcept
- virtual bool equivalent (const error_code &__code, int __i) const noexcept
- virtual bool equivalent (int __i, const error_condition &__cond) const noexcept
- virtual string message (int) const =0
- virtual const char * name () const noexcept=0
- strong_ordering operator<=> (const error_category &__rhs) const noexcept
- error_category & **operator=** (const error_category &)=delete
- bool operator== (const error_category &__other) const noexcept

5.414.1 Detailed Description

Abstract base class for types defining a category of error codes.
An error category defines a context that gives meaning to the integer stored in an `error_code` or `error_condition` object. For example, the standard `errno` constants such a `EINVAL` and `ENOMEM` are associated with the "generic" category and other OS-specific error numbers are associated with the "system" category, but a user-defined category might give different meanings to the same numerical values.
A user-defined category can override the `equivalent` member functions to define correspondence between errors in different categories. For example, a category for errors from disk I/O could consider some of its error numbers equivalent to ENOSPC and ENOENT in the generic category.

Since

C++11

5.414.2 Member Function Documentation

default_error_condition()

```
virtual error_condition std::error_category::default_error_condition (
            int __i) const  [virtual], [noexcept]
```

Return an error_condition corresponding to `i` in this category.

equivalent() [1/2]

```
virtual bool std::error_category::equivalent (
            const error_code & __code,
            int __i) const  [virtual], [noexcept]
```

Test whether `code` corresponds to `i` for this category.

equivalent() [2/2]

```
virtual bool std::error_category::equivalent (
            int __i,
            const error_condition & __cond) const  [virtual], [noexcept]
```

Test whether `cond` corresponds to `i` for this category.

message()

```
virtual string std::error_category::message (
            int ) const  [pure virtual]
```

A description of the error condition corresponding to the number.

name()

```
virtual const char * std::error_category::name () const  [pure virtual], [noexcept]
```

A string that identifies the error category.

operator<=>()

```
strong_ordering std::error_category::operator<=> (
            const error_category & __rhs) const  [inline], [nodiscard], [noexcept]
```

Ordered comparison that defines a total order for error categories.

operator==()

```
bool std::error_category::operator== (
            const error_category & __other) const  [inline], [noexcept]
```

An error_category only compares equal to itself.

The documentation for this class was generated from the following file:

- system_error

5.415 std::error_code Class Reference

`#include <system_error>`

Public Member Functions

- template<typename _ErrorCodeEnum, typename = _Check<_ErrorCodeEnum>>
  error_code (_ErrorCodeEnum __e) noexcept
- **error_code** (const error_code &)=default
- **error_code** (int __v, const error_category &__cat) noexcept
- void **assign** (int __v, const error_category &__cat) noexcept
- const error_category & category () const noexcept
- void **clear** () noexcept
- error_condition default_error_condition () const noexcept
- string message () const
- operator bool () const noexcept
- error_code & **operator=** (const error_code &)=default
- int value () const noexcept

Related Symbols

(Note that these are not member symbols.)

- error_code make_error_code (errc __e) noexcept
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const error_code &__e)
- strong_ordering operator<=> (const error_code &__lhs, const error_code &__rhs) noexcept

5.415.1 Detailed Description

Class error_code
This class is a value type storing an integer error number and a category that gives meaning to the error number. Typically this is done close the the point where the error happens, to capture the original error value.
An `error_code` object can be used to store the original error value emitted by some subsystem, with a category relevant to the subsystem. For example, errors from POSIX library functions can be represented by an `errno` value and the "generic" category, but errors from an HTTP library might be represented by an HTTP response status code (e.g. 404) and a custom category defined by the library.

Since

C++11

5.415.2 Constructor & Destructor Documentation

error_code()

```
template<typename _ErrorCodeEnum, typename = _Check<_ErrorCodeEnum>>
std::error_code::error_code (
            _ErrorCodeEnum __e)  [inline], [noexcept]
```

Initialize with a user-defined type, by calling make_error_code.

5.415.3 Member Function Documentation

category()

```
const error_category & std::error_code::category () const  [inline], [noexcept]
```

The error category that this error belongs to.

default_error_condition()

`error_condition std::error_code::default_error_condition () const [noexcept]`
An `error_condition` for this error's category and value.

message()

`string std::error_code::message () const [inline]`
The category's description of the value.

operator bool()

`std::error_code::operator bool () const [inline], [explicit], [noexcept]`
Test whether `value()` is non-zero.

value()

`int std::error_code::value () const [inline], [noexcept]`
The error value.
The documentation for this class was generated from the following file:

- system_error

5.416 std::error_condition Class Reference

`#include <system_error>`

Public Member Functions

- error_condition () noexcept
- template<typename _ErrorConditionEnum, typename = _Check<_ErrorConditionEnum>>
  error_condition (_ErrorConditionEnum __e) noexcept
- **error_condition** (const error_condition &)=default
- error_condition (int __v, const error_category &__cat) noexcept
- void assign (int __v, const error_category &__cat) noexcept
- const error_category & category () const noexcept
- void clear () noexcept
- string message () const
- operator bool () const noexcept
- error_condition & **operator=** (const error_condition &)=default
- int value () const noexcept

Related Symbols

(Note that these are not member symbols.)

- error_condition make_error_condition (errc __e) noexcept
- strong_ordering operator<=> (const error_condition &__lhs, const error_condition &__rhs) noexcept
- bool operator== (const error_code &__lhs, const error_code &__rhs) noexcept
- bool operator== (const error_code &__lhs, const error_condition &__rhs) noexcept
- bool operator== (const error_condition &__lhs, const error_condition &__rhs) noexcept

5.416.1 Detailed Description

Class error_condition
This class represents error conditions that may be visible at an API boundary. Different `error_code` values that can occur within a library or module might map to the same `error_condition`.
An `error_condition` represents something that the program can test for, and subsequently take appropriate action.

Since

C++11

5.416.2 Constructor & Destructor Documentation

error_condition() [1/3]

```
std::error_condition::error_condition ()  [inline], [noexcept]
```

Initialize with a zero (no error) value and the generic category.

error_condition() [2/3]

```
std::error_condition::error_condition (
            int __v,
            const error_category & __cat)  [inline], [noexcept]
```

Initialize with the specified value and category.

error_condition() [3/3]

```
template<typename _ErrorConditionEnum, typename = _Check<_ErrorConditionEnum>>
std::error_condition::error_condition (
            _ErrorConditionEnum __e)  [inline], [noexcept]
```

Initialize with a user-defined type, by calling make_error_condition.

5.416.3 Member Function Documentation

assign()

```
void std::error_condition::assign (
            int __v,
            const error_category & __cat)  [inline], [noexcept]
```

Set the value and category.

category()

```
const error_category & std::error_condition::category () const  [inline], [noexcept]
```

The error category that this error belongs to.

clear()

```
void std::error_condition::clear ()  [inline], [noexcept]
```

Reset the value and category to the default-constructed state.

message()

```
string std::error_condition::message () const  [inline]
```

The category's description of the value.

operator bool()

```
std::error_condition::operator bool () const  [inline], [explicit], [noexcept]
```
Test whether `value()` is non-zero.

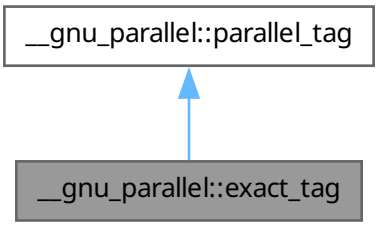
value()

```
int std::error_condition::value () const  [inline], [noexcept]
```
The error value.
The documentation for this class was generated from the following file:

- system_error

5.417 __gnu_parallel::exact_tag Struct Reference

```
#include <tags.h>
```
Inheritance diagram for __gnu_parallel::exact_tag:



Public Member Functions

- **exact_tag** (_ThreadIndex __num_threads)
- _ThreadIndex __get_num_threads ()
- void set_num_threads (_ThreadIndex __num_threads)

5.417.1 Detailed Description

Forces parallel merging with exact splitting, at compile time.

5.417.2 Member Function Documentation

__get_num_threads()

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ()  [inline], [inherited]
```
Find out desired number of threads.

Returns

Desired number of threads.

Referenced by __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), and __gnu_parallel::__parallel_sort(

set_num_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
            _ThreadIndex __num_threads)  [inline], [inherited]
```

Set the desired number of threads.

Parameters

__num_threads	Desired number of threads.

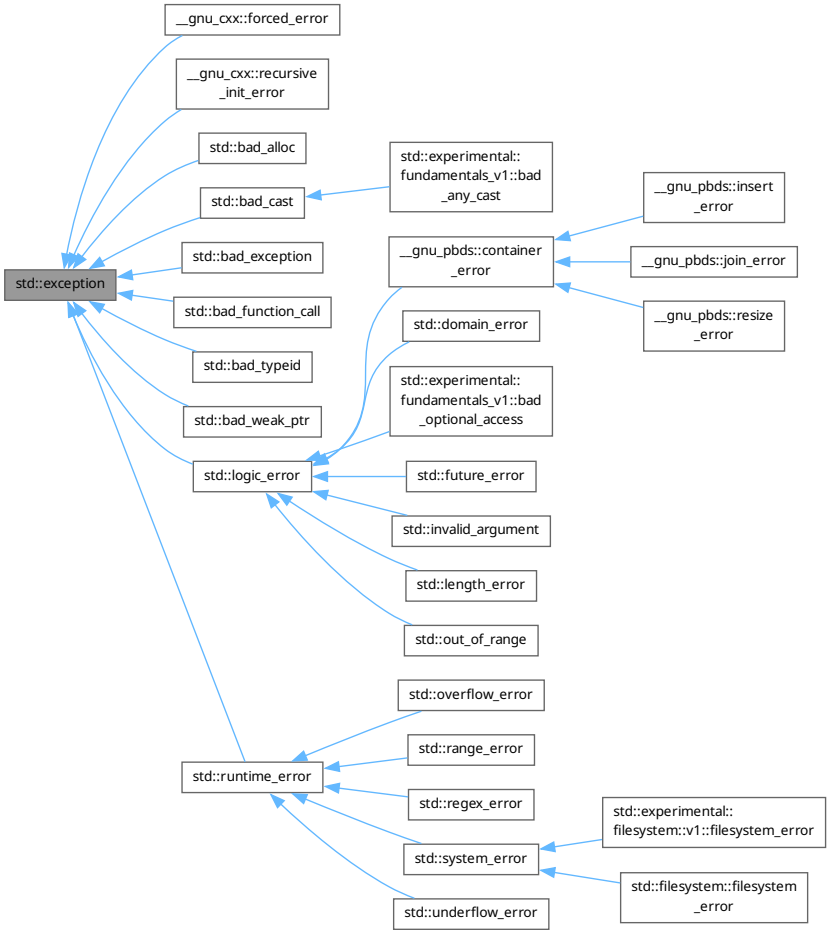
The documentation for this struct was generated from the following file:

- tags.h

5.418 std::exception Class Reference

`#include <exception.h>`

Inheritance diagram for std::exception:



Public Member Functions

- _GLIBCXX26_CONSTEXPR **exception** (const exception &)=default

- _GLIBCXX26_CONSTEXPR **exception** (exception &&)=default
- _GLIBCXX26_CONSTEXPR exception & **operator=** (const exception &)=default
- _GLIBCXX26_CONSTEXPR exception & **operator=** (exception &&)=default
- virtual const char * what () const noexcept

5.418.1 Detailed Description

Base class for all library exceptions.
This is the base class for all exceptions thrown by the standard library, and by certain language expressions. You are free to derive your own exception classes, or use a different hierarchy, or to throw non-class data (e.g., fundamental types).

5.418.2 Member Function Documentation

what()

```
virtual const char * std::exception::what () const  [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error.
Reimplemented in std::bad_alloc, std::bad_cast, std::bad_exception, std::bad_function_call, std::bad_typeid, std::bad_weak_ptr, std::experimental::filesystem::v1::filesystem_error, std::experimental::fundamentals_v1::bad_any_cast, std::filesystem::filesystem_error, std::future_error, std::logic_error, and std::runtime_error.
The documentation for this class was generated from the following file:

- exception.h

5.419 std::__unspecified__::exception_ptr Class Reference

```
#include <exception>
```

Public Member Functions

- **exception_ptr** (const exception_ptr &) noexcept
- **exception_ptr** (exception_ptr &&__o) noexcept
- **exception_ptr** (nullptr_t) noexcept
- const class std::type_info * **__cxa_exception_type** () const noexcept
- **operator bool** () const noexcept
- exception_ptr & **operator=** (const exception_ptr &) noexcept
- exception_ptr & **operator=** (exception_ptr &&__o) noexcept
- void **swap** (exception_ptr &) noexcept

Friends

- bool **operator==** (const exception_ptr &, const exception_ptr &) noexcept=default
- exception_ptr **std::current_exception** () noexcept
- template<typename _Ex>
  exception_ptr **std::make_exception_ptr** (_Ex) noexcept
- void **std::rethrow_exception** (exception_ptr)

Related Symbols

(Note that these are not member symbols.)

- void **swap** (exception_ptr &__lhs, exception_ptr &__rhs)

5.419.1 Detailed Description

An opaque pointer to an arbitrary exception.
The actual name of this type is unspecified, so the alias `std::exception_ptr` should be used to refer to it.

Since

C++11 (but usable in C++98 as a GCC extension)

The documentation for this class was generated from the following file:

- exception_ptr.h

5.420 std::exception_ptr Class Reference

`#include <exception>`

Public Member Functions

- **exception_ptr** (const exception_ptr &) noexcept
- **exception_ptr** (exception_ptr &&__o) noexcept
- **exception_ptr** (nullptr_t) noexcept
- const class std::type_info * **__cxa_exception_type** () const noexcept
- **operator bool** () const noexcept
- exception_ptr & **operator=** (const exception_ptr &) noexcept
- exception_ptr & **operator=** (exception_ptr &&__o) noexcept
- void **swap** (exception_ptr &) noexcept

Friends

- bool **operator==** (const exception_ptr &, const exception_ptr &) noexcept=default
- exception_ptr **std::current_exception** () noexcept
- template<typename _Ex>
  exception_ptr **std::make_exception_ptr** (_Ex) noexcept
- void **std::rethrow_exception** (exception_ptr)

Related Symbols

(Note that these are not member symbols.)

- void **swap** (exception_ptr &__lhs, exception_ptr &__rhs)

5.420.1 Detailed Description

An opaque pointer to an arbitrary exception.
The actual name of this type is unspecified, so the alias `std::exception_ptr` should be used to refer to it.

Since

C++11 (but usable in C++98 as a GCC extension)

The documentation for this class was generated from the following file:

- exception_ptr.h

5.421 std::exponential_distribution< _RealType > Class Template Reference

`#include <random>`

Classes

- struct param_type

Public Types

- typedef _RealType result_type

Public Member Functions

- exponential_distribution ()
- exponential_distribution (_RealType __lambda)
- **exponential_distribution** (const param_type &__p)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- _RealType lambda () const
- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- bool operator== (const exponential_distribution &__d1, const exponential_distribution &__d2)

5.421.1 Detailed Description

template<typename _RealType = double>
class std::exponential_distribution< _RealType >

An exponential continuous distribution for random numbers.
The formula for the exponential probability density function is $p(x|\lambda) = \lambda e^{-\lambda x}$.

Table 2344 Distribution Statistics

Mean	$\frac{1}{\lambda}$
Median	$\frac{\ln 2}{\lambda}$
Mode	$zero$
Range	$[0, \infty]$
Standard Deviation	$\frac{1}{\lambda}$

Since

C++11

5.421.2 Member Typedef Documentation

result_type

```
template<typename _RealType = double>
typedef _RealType std::exponential_distribution< _RealType >::result_type
```

The type of the range of the distribution.

5.421.3 Constructor & Destructor Documentation

exponential_distribution() [1/2]

```
template<typename _RealType = double>
std::exponential_distribution< _RealType >::exponential_distribution ()  [inline]
```

Constructs an exponential distribution with inverse scale parameter 1.0.

exponential_distribution() [2/2]

```
template<typename _RealType = double>
std::exponential_distribution< _RealType >::exponential_distribution (
            _RealType __lambda)  [inline], [explicit]
```

Constructs an exponential distribution with inverse scale parameter λ.

5.421.4 Member Function Documentation

lambda()

```
template<typename _RealType = double>
_RealType std::exponential_distribution< _RealType >::lambda () const  [inline]
```

Returns the inverse scale parameter of the distribution.

max()

```
template<typename _RealType = double>
result_type std::exponential_distribution< _RealType >::max () const  [inline]
```

Returns the least upper bound value of the distribution.

min()

```
template<typename _RealType = double>
result_type std::exponential_distribution< _RealType >::min () const  [inline]
```

Returns the greatest lower bound value of the distribution.

operator()()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::exponential_distribution< _RealType >::operator() (
            _UniformRandomNumberGenerator & __urng)  [inline]
```

Generating functions.
Referenced by operator()().

param() [1/2]

```
template<typename _RealType = double>
param_type std::exponential_distribution< _RealType >::param () const  [inline]
```

Returns the parameter set of the distribution.
Referenced by std::operator>>().

param() [2/2]

```
template<typename _RealType = double>
void std::exponential_distribution< _RealType >::param (
            const param_type & __param)  [inline]
```

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

reset()

```
template<typename _RealType = double>
void std::exponential_distribution< _RealType >::reset ()  [inline]
```

Resets the distribution state.
Has no effect on exponential distributions.

5.421.5 Friends And Related Symbol Documentation

operator==

```
template<typename _RealType = double>
bool operator== (
            const exponential_distribution< _RealType > & __d1,
            const exponential_distribution< _RealType > & __d2)  [friend]
```

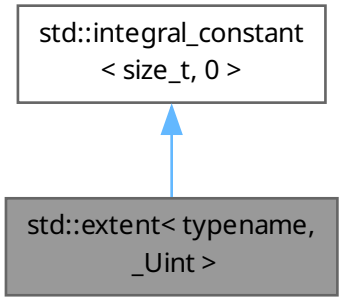
Return true if two exponential distributions have the same parameters.
The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

5.422 std::extent< typename, _Uint > Struct Template Reference

```
#include <type_traits>
```

Inheritance diagram for std::extent< typename, _Uint >:



Public Types

- using **type**
- using **value_type**

Public Member Functions

- constexpr **operator value_type** () const noexcept

Static Public Attributes

- static constexpr size_t **value**

5.422.1 Detailed Description

template<typename, unsigned _Uint = 0>
struct std::extent< typename, _Uint >

extent
The documentation for this struct was generated from the following file:

- type_traits

5.423 std::extreme_value_distribution< _RealType > Class Template Reference

`#include <random>`

Classes

- struct param_type

Public Types

- typedef _RealType result_type

Public Member Functions

- **extreme_value_distribution** (_RealType __a, _RealType __b=_RealType(1))
- **extreme_value_distribution** (const param_type &__p)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- _RealType a () const
- _RealType b () const
- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- bool operator== (const extreme_value_distribution &__d1, const extreme_value_distribution &__d2)

5.423.1 Detailed Description

template<typename _RealType = double>
class std::extreme_value_distribution< _RealType >

A extreme_value_distribution random number distribution.
The formula for the normal probability mass function is

$$p(x|a,b) = \frac{1}{b}\exp(\frac{a-x}{b} - \exp(\frac{a-x}{b}))$$

Since

C++11

5.423.2 Member Typedef Documentation

result_type

```
template<typename _RealType = double>
typedef _RealType std::extreme_value_distribution< _RealType >::result_type
```

The type of the range of the distribution.

5.423.3 Member Function Documentation

a()

```
template<typename _RealType = double>
_RealType std::extreme_value_distribution< _RealType >::a () const  [inline]
```

Return the a parameter of the distribution.

b()

```
template<typename _RealType = double>
_RealType std::extreme_value_distribution< _RealType >::b () const  [inline]
```

Return the b parameter of the distribution.

max()

```
template<typename _RealType = double>
result_type std::extreme_value_distribution< _RealType >::max () const  [inline]
```

Returns the least upper bound value of the distribution.

min()

```
template<typename _RealType = double>
result_type std::extreme_value_distribution< _RealType >::min () const  [inline]
```

Returns the greatest lower bound value of the distribution.

operator()()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::extreme_value_distribution< _RealType >::operator() (
            _UniformRandomNumberGenerator & __urng)  [inline]
```

Generating functions.
Referenced by operator()().

param() [1/2]

```
template<typename _RealType = double>
param_type std::extreme_value_distribution< _RealType >::param () const  [inline]
```

Returns the parameter set of the distribution.
Referenced by std::operator>>().

param() [2/2]

```
template<typename _RealType = double>
void std::extreme_value_distribution< _RealType >::param (
            const param_type & __param)  [inline]
```

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

reset()

```
template<typename _RealType = double>
void std::extreme_value_distribution< _RealType >::reset ()  [inline]
```

Resets the distribution state.

5.423.4 Friends And Related Symbol Documentation

operator==

```
template<typename _RealType = double>
bool operator== (
            const extreme_value_distribution< _RealType > & __d1,
            const extreme_value_distribution< _RealType > & __d2)  [friend]
```

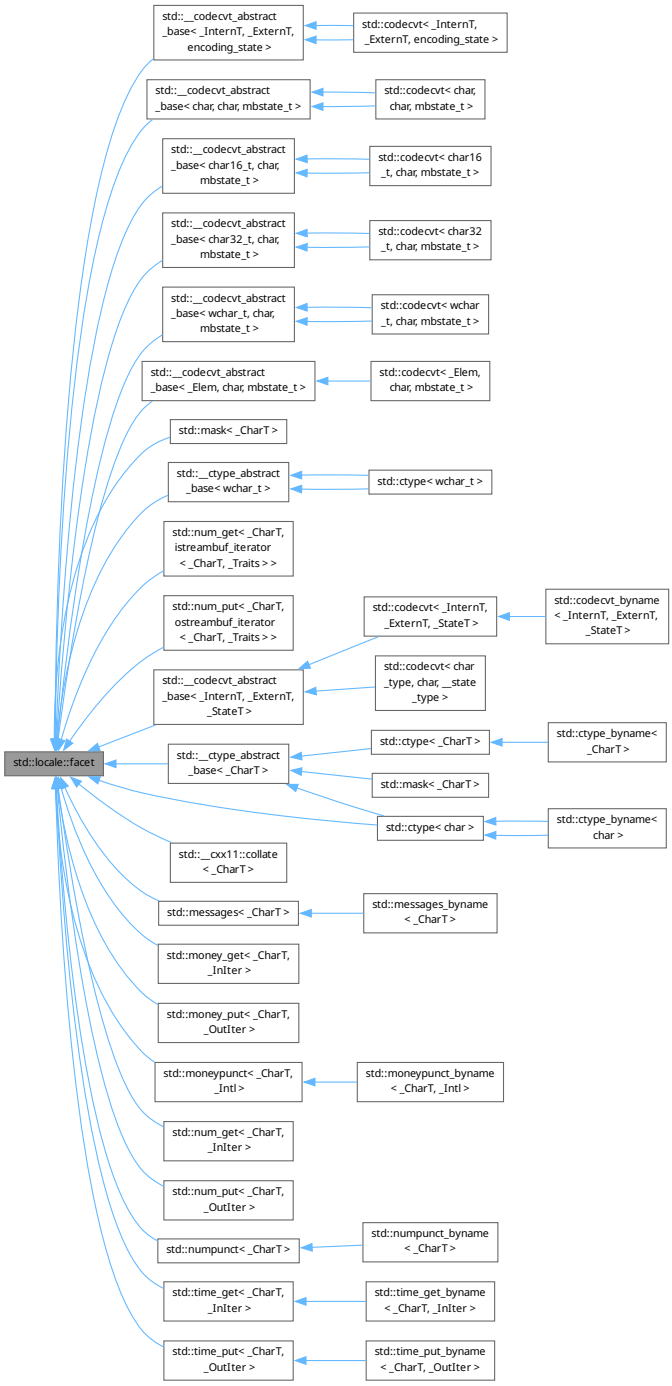
Return true if two extreme value distributions have the same parameters.
The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

5.424 std::locale::facet Class Reference

```
#include <locale_classes.h>
```

Inheritance diagram for std::locale::facet:



Protected Member Functions

- **facet** (const facet &)=delete
- facet (size_t __refs=0) throw ()

- virtual ~facet ()
- facet & **operator=** (const facet &)=delete

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Friends

- class **locale**
- class **locale::_Impl**

5.424.1 Detailed Description

Localization functionality base class.
The facet class is the base class for a localization feature, such as money, time, and number printing. It provides common support for facets and reference management.
Facets may not be copied or assigned.

5.424.2 Constructor & Destructor Documentation

facet()

```
std::locale::facet::facet (
            size_t __refs = 0) throw ( )   [inline], [explicit], [protected]
```

Facet constructor.
This is the constructor provided by the standard. If refs is 0, the facet is destroyed when the last referencing locale is destroyed. Otherwise the facet will never be destroyed.

Parameters

__refs	The initial value for reference count.

Referenced by std::__cxx11::collate< _CharT >::collate(), std::__cxx11::collate< _CharT >::collate(), std::messages< _CharT >::messa
std::messages< _CharT >::messages(), std::money_get< _CharT, _InIter >::money_get(), std::money_put< _CharT, _OutIter >::money
std::moneypunct< _CharT, _Intl >::moneypunct(), std::moneypunct< _CharT, _Intl >::moneypunct(), std::moneypunct< _CharT, _Intl >:
std::num_get< _CharT, istreambuf_iterator< _CharT, _Traits > >::num_get(), std::num_put< _CharT, ostreambuf_iterator< _CharT, _Tr
std::numpunct< _CharT >::numpunct(), std::numpunct< _CharT >::numpunct(), std::numpunct< _CharT >::numpunct(),
std::time_get< _CharT, _InIter >::time_get(), std::time_put< _CharT, _OutIter >::time_put(), and ~facet().

~facet()

```
virtual std::locale::facet::~facet ()  [protected], [virtual]
```

Facet destructor.
References facet().
The documentation for this class was generated from the following file:

- locale_classes.h

5.425 std::filesystem::file_status Class Reference

`#include <filesystem>`

Public Member Functions

- **file_status** (const file_status &) noexcept=default
- **file_status** (file_status &&) noexcept=default
- **file_status** (file_type __ft, perms __prms=perms::unknown) noexcept
- file_status & **operator=** (const file_status &) noexcept=default
- file_status & **operator=** (file_status &&) noexcept=default
- perms **permissions** () const noexcept
- void **permissions** (perms __prms) noexcept
- file_type **type** () const noexcept
- void **type** (file_type __ft) noexcept

Friends

- bool **operator==** (const file_status &, const file_status &) noexcept=default

5.425.1 Detailed Description

Information about a file's type and permissions.

Since

C++17

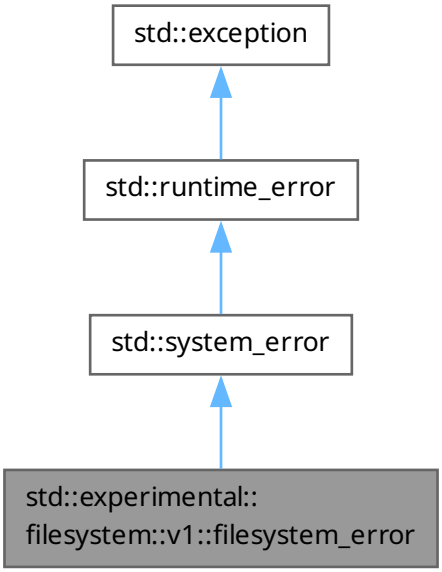
The documentation for this class was generated from the following file:

- bits/fs_dir.h

5.426 std::experimental::filesystem::v1::filesystem_error Class Reference

`#include <fs_path.h>`

Inheritance diagram for std::experimental::filesystem::v1::filesystem_error:



Public Member Functions

- **filesystem_error** (const string &__what_arg, const path &__p1, const path &__p2, error_code __ec)
- **filesystem_error** (const string &__what_arg, const path &__p1, error_code __ec)
- **filesystem_error** (const string &__what_arg, error_code __ec)
- const error_code & **code** () const noexcept
- const path & **path1** () const noexcept
- const path & **path2** () const noexcept
- const char * what () const noexcept

5.426.1 Detailed Description

Exception type thrown by the Filesystem TS library.

5.426.2 Member Function Documentation

what()

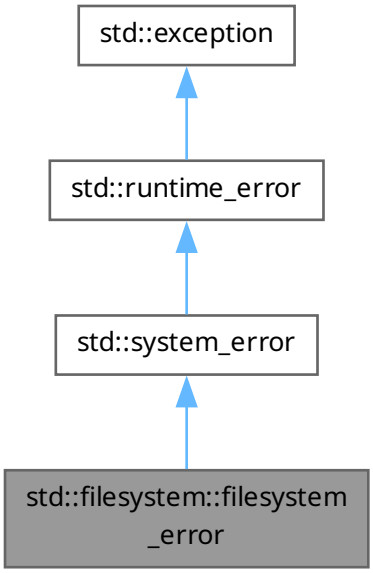
```
const char * std::experimental::filesystem::v1::filesystem_error::what () const  [inline], [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).
Reimplemented from std::runtime_error.
The documentation for this class was generated from the following file:

- experimental/bits/fs_path.h

5.427 std::filesystem::filesystem_error Class Reference

`#include <filesystem>`
Inheritance diagram for std::filesystem::filesystem_error:



Public Member Functions

- **filesystem_error** (const filesystem_error &)=default
- **filesystem_error** (const string &__what_arg, const path &__p1, const path &__p2, error_code __ec)
- **filesystem_error** (const string &__what_arg, const path &__p1, error_code __ec)
- **filesystem_error** (const string &__what_arg, error_code __ec)
- const error_code & **code** () const noexcept
- filesystem_error & **operator=** (const filesystem_error &)=default
- const path & **path1** () const noexcept
- const path & **path2** () const noexcept
- const char * what () const noexcept

5.427.1 Detailed Description

Exception type thrown by the Filesystem library.

Since

C++17

5.427.2 Member Function Documentation

what()

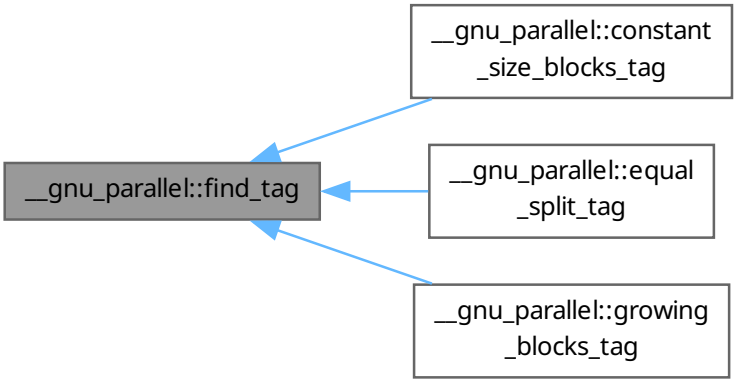
```
const char * std::filesystem::filesystem_error::what () const  [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).
Reimplemented from std::runtime_error.
References std::make_error_code().
The documentation for this class was generated from the following file:

- bits/fs_path.h

5.428 __gnu_parallel::find_tag Struct Reference

`#include <tags.h>`
Inheritance diagram for __gnu_parallel::find_tag:



5.428.1 Detailed Description

Base class for for std::find() variants.
The documentation for this struct was generated from the following file:

- tags.h

5.429 std::fisher_f_distribution< _RealType > Class Template Reference

`#include <random>`

Classes

- struct param_type

Public Types

- typedef _RealType result_type

Public Member Functions

- **fisher_f_distribution** (_RealType __m, _RealType __n=_RealType(1))
- **fisher_f_distribution** (const param_type &__p)

- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- _RealType **m** () const
- result_type max () const
- result_type min () const
- _RealType **n** () const
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- template<typename _RealType1, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::fisher_f_distribution< _RealType1 > &__x)
- bool operator== (const fisher_f_distribution &__d1, const fisher_f_distribution &__d2)
- template<typename _RealType1, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::fisher_f_distribution< _RealType1 > &__x)

5.429.1 Detailed Description

template<typename _RealType = double>
class std::fisher_f_distribution< _RealType >

A fisher_f_distribution random number distribution.
The formula for the normal probability mass function is

$$p(x|m,n) = \frac{\Gamma((m+n)/2)}{\Gamma(m/2)\Gamma(n/2)}(\frac{m}{n})^{m/2}x^{(m/2)-1}(1+\frac{mx}{n})^{-(m+n)/2}$$

Since

C++11

5.429.2 Member Typedef Documentation

result_type

```
template<typename _RealType = double>
typedef _RealType std::fisher_f_distribution< _RealType >::result_type
```

The type of the range of the distribution.

5.429.3 Member Function Documentation

max()

```
template<typename _RealType = double>
result_type std::fisher_f_distribution< _RealType >::max () const  [inline]
```

Returns the least upper bound value of the distribution.

min()

```
template<typename _RealType = double>
result_type std::fisher_f_distribution< _RealType >::min () const  [inline]
```

Returns the greatest lower bound value of the distribution.

operator()()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::fisher_f_distribution< _RealType >::operator() (
            _UniformRandomNumberGenerator & __urng)  [inline]
```

Generating functions.

param() [1/2]

```
template<typename _RealType = double>
param_type std::fisher_f_distribution< _RealType >::param () const  [inline]
```

Returns the parameter set of the distribution.

param() [2/2]

```
template<typename _RealType = double>
void std::fisher_f_distribution< _RealType >::param (
            const param_type & __param)  [inline]
```

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

reset()

```
template<typename _RealType = double>
void std::fisher_f_distribution< _RealType >::reset ()  [inline]
```

Resets the distribution state.

5.429.4 Friends And Related Symbol Documentation

operator<<

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::fisher_f_distribution< _RealType1 > & __x)  [friend]
```

Inserts a fisher_f_distribution random number distribution __x into the output stream __os.

Parameters

__os	An output stream.
__x	A fisher_f_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

operator==

```
template<typename _RealType = double>
bool operator== (
             const fisher_f_distribution< _RealType > & __d1,
             const fisher_f_distribution< _RealType > & __d2)  [friend]
```

Return true if two Fisher f distributions have the same parameters and the sequences that would be generated are equal.

operator>>

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
             std::basic_istream< _CharT, _Traits > & __is,
             std::fisher_f_distribution< _RealType1 > & __x)  [friend]
```

Extracts a fisher_f_distribution random number distribution __x from the input stream __is.

Parameters

_↩ _is	An input stream.
_↩ _x	A fisher_f_distribution random number generator engine.

Returns

The input stream with __x extracted or in an error state.

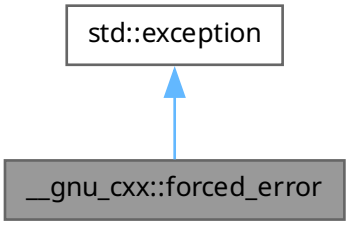
The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

5.430 __gnu_cxx::forced_error Struct Reference

```
#include <throw_allocator.h>
```

Inheritance diagram for __gnu_cxx::forced_error:



Public Member Functions

- virtual const char * what () const noexcept

5.430.1 Detailed Description

Thrown by utilities for testing exception safety.

5.430.2 Member Function Documentation

what()

```
virtual const char * std::exception::what () const  [virtual], [noexcept], [inherited]
```

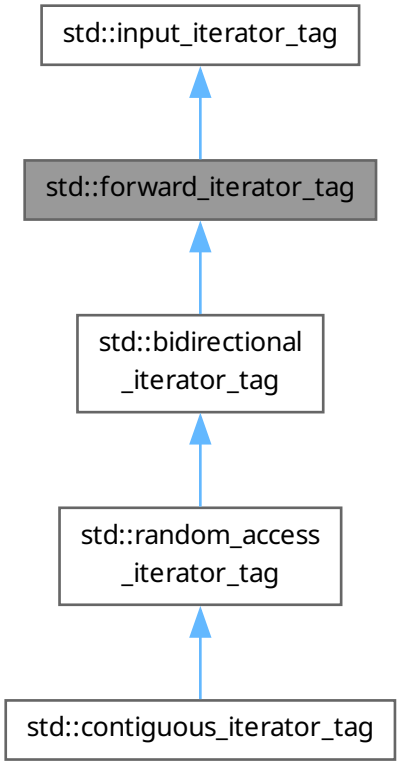
Returns a C-style character string describing the general cause of the current error.
Reimplemented in std::bad_alloc, std::bad_cast, std::bad_exception, std::bad_function_call, std::bad_typeid, std::bad_weak_ptr, std::experimental::filesystem::v1::filesystem_error, std::experimental::fundamentals_v1::bad_any_cast, std::filesystem::filesystem_error, std::future_error, std::logic_error, and std::runtime_error.
The documentation for this struct was generated from the following file:

- throw_allocator.h

5.431 std::forward_iterator_tag Struct Reference

```
#include <stl_iterator_base_types.h>
```

Inheritance diagram for std::forward_iterator_tag:



5.431.1 Detailed Description

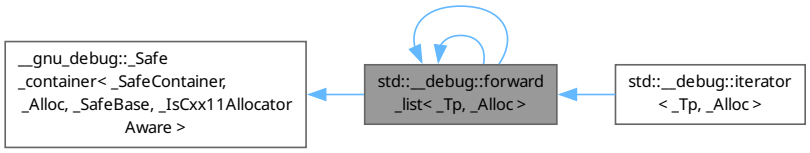
Forward iterators support a superset of input iterator operations.
The documentation for this struct was generated from the following file:

- stl_iterator_base_types.h

5.432 std::__debug::forward_list< _Tp, _Alloc > Class Template Reference

`#include <forward_list>`

Inheritance diagram for std::__debug::forward_list< _Tp, _Alloc >:



Public Types

- typedef _Base::allocator_type **allocator_type**
- typedef __gnu_debug::_Safe_iterator< _Base_const_iterator, forward_list > **const_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**
- typedef _Base::difference_type **difference_type**
- typedef __gnu_debug::_Safe_iterator< _Base_iterator, forward_list > **iterator**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef _Base::size_type **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **forward_list** (_Base_ref __x)
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  forward_list (_InputIterator __first, _InputIterator __last, const allocator_type &__al=allocator_type())
- **forward_list** (const allocator_type &__al) noexcept
- **forward_list** (const forward_list &)=default
- **forward_list** (const forward_list &__list, const allocator_type &__al)
- **forward_list** (forward_list &&)=default
- **forward_list** (forward_list &&__list, const allocator_type &__al) noexcept(std::is_nothrow_constructible< _Base, _Base, const allocator_type & >::value)
- **forward_list** (size_type __n, const __type_identity_t< _Tp > &__value, const allocator_type &__al=allocator_type())
- **forward_list** (size_type __n, const allocator_type &__al=allocator_type())
- **forward_list** (std::initializer_list< _Tp > __il, const allocator_type &__al=allocator_type())
- const _Base & **_M_base** () const noexcept
- _Base & **_M_base** () noexcept
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  void **assign** (_InputIterator __first, _InputIterator __last)
- void **assign** (size_type __n, const _Tp &__val)
- void **assign** (std::initializer_list< _Tp > __il)
- const_iterator **before_begin** () const noexcept
- iterator **before_begin** () noexcept
- const_iterator **begin** () const noexcept
- iterator **begin** () noexcept
- const_iterator **cbefore_begin** () const noexcept
- const_iterator **cbegin** () const noexcept

- const_iterator **cend** () const noexcept
- void **clear** () noexcept
- template<typename... _Args>
  iterator **emplace_after** (const_iterator __pos, _Args &&... __args)
- const_iterator **end** () const noexcept
- iterator **end** () noexcept
- iterator **erase_after** (const_iterator __pos)
- iterator **erase_after** (const_iterator __pos, const_iterator __last)
- reference **front** ()
- const_reference **front** () const
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  iterator **insert_after** (const_iterator __pos, _InputIterator __first, _InputIterator __last)
- iterator **insert_after** (const_iterator __pos, _Tp &&__val)
- iterator **insert_after** (const_iterator __pos, const _Tp &__val)
- iterator **insert_after** (const_iterator __pos, size_type __n, const _Tp &__val)
- iterator **insert_after** (const_iterator __pos, std::initializer_list< _Tp > __il)
- void **merge** (forward_list &&__list)
- template<typename _Comp>
  void **merge** (forward_list &&__list, _Comp __comp)
- void **merge** (forward_list &__list)
- template<typename _Comp>
  void **merge** (forward_list &__list, _Comp __comp)
- forward_list & **operator=** (const forward_list &)=default
- forward_list & **operator=** (forward_list &&)=default
- forward_list & **operator=** (std::initializer_list< _Tp > __il)
- void **pop_front** ()
- __remove_return_type **remove** (const _Tp &__val)
- template<typename _Pred>
  __remove_return_type **remove_if** (_Pred __pred)
- void **resize** (size_type __sz)
- void **resize** (size_type __sz, const value_type &__val)
- void **splice_after** (const_iterator __pos, forward_list &&__list)
- void **splice_after** (const_iterator __pos, forward_list &&__list, const_iterator __before, const_iterator __last)
- void **splice_after** (const_iterator __pos, forward_list &&__list, const_iterator __i)
- void **splice_after** (const_iterator __pos, forward_list &__list)
- void **splice_after** (const_iterator __pos, forward_list &__list, const_iterator __before, const_iterator __last)
- void **splice_after** (const_iterator __pos, forward_list &__list, const_iterator __i)
- void **swap** (forward_list &__list) noexcept(noexcept(declval< _Base & >().swap(__list)))
- __remove_return_type **unique** ()
- template<typename _BinPred>
  __remove_return_type **unique** (_BinPred __binary_pred)

Protected Member Functions

- constexpr void **_M_swap** (const _Safe_container &__x) const noexcept

Friends

- template<typename _ItT, typename _SeqT, typename _CatT>
  class **::__gnu_debug::_Safe_iterator**

5.432.1 Detailed Description

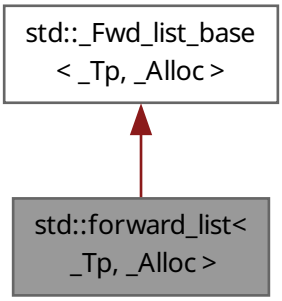
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
class std::__debug::forward_list< _Tp, _Alloc >

Class std::forward_list with safety/checking/debug instrumentation.
The documentation for this class was generated from the following file:

- debug/forward_list

5.433 std::forward_list< _Tp, _Alloc > Class Template Reference

`#include <forward_list>`
Inheritance diagram for std::forward_list< _Tp, _Alloc >:



Public Types

- typedef _Alloc **allocator_type**
- typedef _Base::const_iterator **const_iterator**
- typedef _Alloc_traits::const_pointer **const_pointer**
- typedef const value_type & **const_reference**
- typedef std::ptrdiff_t **difference_type**
- typedef _Base::iterator **iterator**
- typedef _Alloc_traits::pointer **pointer**
- typedef value_type & **reference**
- typedef std::size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- forward_list ()=default
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  forward_list (_InputIterator __first, _InputIterator __last, const _Alloc &__al=_Alloc())
- forward_list (const _Alloc &__al) noexcept
- forward_list (const forward_list &__list)
- forward_list (const forward_list &__list, const __type_identity_t< _Alloc > &__al)
- forward_list (forward_list &&)=default

- forward_list (forward_list &&__list, const __type_identity_t< _Alloc > &__al) noexcept(_Node_alloc_traits::_S↩ _always_equal())
- forward_list (size_type __n, const _Alloc &__al=_Alloc())
- forward_list (size_type __n, const _Tp &__value, const _Alloc &__al=_Alloc())
- forward_list (std::initializer_list< _Tp > __il, const _Alloc &__al=_Alloc())
- ~forward_list () noexcept
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  void assign (_InputIterator __first, _InputIterator __last)
- void assign (size_type __n, const _Tp &__val)
- void assign (std::initializer_list< _Tp > __il)
- const_iterator before_begin () const noexcept
- iterator before_begin () noexcept
- const_iterator begin () const noexcept
- iterator begin () noexcept
- const_iterator cbefore_begin () const noexcept
- const_iterator cbegin () const noexcept
- const_iterator cend () const noexcept
- void clear () noexcept
- template<typename... _Args>
  iterator emplace_after (const_iterator __pos, _Args &&... __args)
- template<typename... _Args>
  reference emplace_front (_Args &&... __args)
- bool empty () const noexcept
- const_iterator end () const noexcept
- iterator end () noexcept
- iterator erase_after (const_iterator __pos)
- iterator erase_after (const_iterator __pos, const_iterator __last)
- reference front ()
- const_reference front () const
- allocator_type get_allocator () const noexcept
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  iterator insert_after (const_iterator __pos, _InputIterator __first, _InputIterator __last)
- iterator **insert_after** (const_iterator __pos, _Tp &&__val)
- iterator insert_after (const_iterator __pos, const _Tp &__val)
- iterator insert_after (const_iterator __pos, size_type __n, const _Tp &__val)
- iterator insert_after (const_iterator __pos, std::initializer_list< _Tp > __il)
- size_type max_size () const noexcept
- void merge (forward_list &&__list)
- template<typename _Comp>
  void merge (forward_list &&__list, _Comp __comp)
- void **merge** (forward_list &__list)
- template<typename _Comp>
  void **merge** (forward_list &__list, _Comp __comp)
- forward_list & operator= (const forward_list &__list)
- forward_list & operator= (forward_list &&__list) noexcept(_Node_alloc_traits::_S_nothrow_move())
- forward_list & operator= (std::initializer_list< _Tp > __il)
- void pop_front ()
- void **push_front** (_Tp &&__val)
- void push_front (const _Tp &__val)
- __remove_return_type remove (const _Tp &__val)

- template<typename _Pred>
  __remove_return_type remove_if (_Pred __pred)
- template<typename _Pred>
  auto **remove_if** (_Pred __pred) -> __remove_return_type
- void resize (size_type __sz)
- void resize (size_type __sz, const value_type &__val)
- void reverse () noexcept
- void sort ()
- template<typename _Comp>
  void sort (_Comp __comp)
- void splice_after (const_iterator __pos, forward_list &&__list) noexcept
- void splice_after (const_iterator __pos, forward_list &&__list, const_iterator __i) noexcept
- void **splice_after** (const_iterator __pos, forward_list &__list) noexcept
- void **splice_after** (const_iterator __pos, forward_list &__list, const_iterator __i) noexcept
- void swap (forward_list &__list) noexcept
- __remove_return_type unique ()
- template<typename _BinPred>
  __remove_return_type unique (_BinPred __binary_pred)
- template<typename _BinPred>
  auto **unique** (_BinPred __binary_pred) -> __remove_return_type

- void splice_after (const_iterator __pos, forward_list &&, const_iterator __before, const_iterator __last) noexcept
- void splice_after (const_iterator __pos, forward_list &, const_iterator __before, const_iterator __last) noexcept

5.433.1 Detailed Description

template<typename _Tp, typename _Alloc = allocator<_Tp>>
class std::forward_list< _Tp, _Alloc >

A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.

Since

C++11

Template Parameters

_Tp	Type of element.
_Alloc	Allocator type, defaults to allocator<_Tp>.

Meets the requirements of a `container`, a `sequence`, including the `optional sequence requirements` with the exception of `at` and `operator[]`.
This is a *singly linked* list. Traversal up the list requires linear time, but adding and removing elements (or *nodes*) is done in constant time, regardless of where the change takes place. Unlike std::vector and std::deque, random-access iterators are not provided, so subscripting (`[]`) access is not allowed. For algorithms which only need sequential access, this lack makes no difference.
Also unlike the other standard containers, std::forward_list provides specialized algorithms unique to linked lists, such as splicing, sorting, and in-place reversal.

5.433.2 Constructor & Destructor Documentation

forward_list() [1/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list ()  [default]
```

Creates a forward_list with no elements.
Referenced by insert_after(), insert_after(), merge(), operator=(), remove(), resize(), resize(), sort(), and splice_after().

forward_list() [2/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
            const _Alloc & __al)  [inline], [explicit], [noexcept]
```

Creates a forward_list with no elements.

Parameters

_↩ _al	An allocator object.

forward_list() [3/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
            const forward_list< _Tp, _Alloc > & __list,
            const __type_identity_t< _Alloc > & __al)  [inline]
```

Copy constructor with allocator argument.

Parameters

__list	Input list to copy.
__al	An allocator object.

forward_list() [4/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
            forward_list< _Tp, _Alloc > && __list,
            const __type_identity_t< _Alloc > & __al)  [inline], [noexcept]
```

Move constructor with allocator argument.

Parameters

__list	Input list to move.
__al	An allocator object.

forward_list() [5/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
```

```
            size_type __n,
            const _Alloc & __al = _Alloc())  [inline], [explicit]
```

Creates a forward_list with default constructed elements.

Parameters

_↩ _n	The number of elements to initially create.
_↩ _al	An allocator object.

This constructor creates the forward_list with __n default constructed elements.

forward_list() [6/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
            size_type __n,
            const _Tp & __value,
            const _Alloc & __al = _Alloc())  [inline]
```

Creates a forward_list with copies of an exemplar element.

Parameters

__n	The number of elements to initially create.
__value	An element to copy.
__al	An allocator object.

This constructor fills the forward_list with __n copies of __value.

forward_list() [7/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
std::forward_list< _Tp, _Alloc >::forward_list (
            _InputIterator __first,
            _InputIterator __last,
            const _Alloc & __al = _Alloc())  [inline]
```

Builds a forward_list from a range.

Parameters

__first	An input iterator.
__last	An input iterator.
__al	An allocator object.

Create a forward_list consisting of copies of the elements from [__first,__last). This is linear in N (where N is distance(__first,__last)).

forward_list() [8/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
            const forward_list< _Tp, _Alloc > & __list)  [inline]
```

The forward_list copy constructor.

Parameters

__list	A forward_list of identical element and allocator types.

forward_list() [9/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
            forward_list< _Tp, _Alloc > && )  [default]
```

The forward_list move constructor.

Parameters

__list	A forward_list of identical element and allocator types.

The newly-created forward_list contains the exact contents of the moved instance. The contents of the moved instance are a valid, but unspecified forward_list.

forward_list() [10/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
            std::initializer_list< _Tp > __il,
            const _Alloc & __al = _Alloc())  [inline]
```

Builds a forward_list from an initializer_list.

Parameters

_↩ _il	An initializer_list of value_type.
_↩ _al	An allocator object.

Create a forward_list consisting of copies of the elements in the initializer_list __il. This is linear in __il.size().

~forward_list()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::~forward_list ()  [inline], [noexcept]
```

The forward_list dtor.

5.433.3 Member Function Documentation

assign() [1/3]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
void std::forward_list< _Tp, _Alloc >::assign (
            _InputIterator __first,
            _InputIterator __last)  [inline]
```

Assigns a range to a forward_list.

Parameters

__first	An input iterator.
__last	An input iterator.

This function fills a forward_list with copies of the elements in the range [__first,__last).
Note that the assignment completely changes the forward_list and that the number of elements of the resulting forward_list is the same as the number of elements assigned.
Referenced by std::forward_list< _Tp, polymorphic_allocator< _Tp > >::assign(), operator=(), std::forward_list< _Tp, polymorphic_alloc and std::forward_list< _Tp, polymorphic_allocator< _Tp > >::operator=().

assign() [2/3]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::assign (
            size_type __n,
            const _Tp & __val)  [inline]
```

Assigns a given value to a forward_list.

Parameters

__n	Number of elements to be assigned.
__val	Value to be assigned.

This function fills a forward_list with __n copies of the given value. Note that the assignment completely changes the forward_list, and that the resulting forward_list has __n elements.

assign() [3/3]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::assign (
            std::initializer_list< _Tp > __il)  [inline]
```

Assigns an initializer_list to a forward_list.

Parameters

__il	An initializer_list of value_type.

Replace the contents of the forward_list with copies of the elements in the initializer_list __il. This is linear in __il.size().

before_begin() [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list< _Tp, _Alloc >::before_begin () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points before the first element in the forward_list. Iteration is done in ordinary element order.

before_begin() [2/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list< _Tp, _Alloc >::before_begin ()  [inline], [noexcept]
```

Returns a read/write iterator that points before the first element in the forward_list. Iteration is done in ordinary element order.
Referenced by std::forward_list< _Tp, polymorphic_allocator< _Tp > >::assign(), std::forward_list< _Tp, polymorphic_allocator< _Tp > insert_after(), insert_after(), resize(), and resize().

begin() [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list< _Tp, _Alloc >::begin () const  [inline], [noexcept]
```
Returns a read-only (constant) iterator that points to the first element in the forward_list. Iteration is done in ordinary element order.

begin() [2/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list< _Tp, _Alloc >::begin ()  [inline], [noexcept]
```
Returns a read/write iterator that points to the first element in the forward_list. Iteration is done in ordinary element order.
Referenced by std::forward_list< _Tp, polymorphic_allocator< _Tp > >::assign(), and std::forward_list< _Tp, polymorphic_allocator< _

cbefore_begin()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list< _Tp, _Alloc >::cbefore_begin () const  [inline], [noexcept]
```
Returns a read-only (constant) iterator that points before the first element in the forward_list. Iteration is done in ordinary element order.
Referenced by std::forward_list< _Tp, polymorphic_allocator< _Tp > >::assign(), std::forward_list< _Tp, polymorphic_allocator< _Tp > std::forward_list< _Tp, polymorphic_allocator< _Tp > >::emplace_front(), std::forward_list< _Tp, polymorphic_allocator< _Tp > >::pus and remove().

cbegin()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list< _Tp, _Alloc >::cbegin () const  [inline], [noexcept]
```
Returns a read-only (constant) iterator that points to the first element in the forward_list. Iteration is done in ordinary element order.
Referenced by std::operator==().

cend()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list< _Tp, _Alloc >::cend () const  [inline], [noexcept]
```
Returns a read-only (constant) iterator that points one past the last element in the forward_list. Iteration is done in ordinary element order.
Referenced by std::operator==().

clear()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::clear ()  [inline], [noexcept]
```
Erases all the elements.
Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.
Referenced by std::forward_list< _Tp, polymorphic_allocator< _Tp > >::assign(), std::forward_list< _Tp, polymorphic_allocator< _Tp > operator=(), and std::forward_list< _Tp, polymorphic_allocator< _Tp > >::operator=().

emplace_after()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
template<typename... _Args>
iterator std::forward_list< _Tp, _Alloc >::emplace_after (
            const_iterator __pos,
            _Args &&... __args)  [inline]
```

Constructs object in forward_list after the specified iterator.

Parameters

__pos	A const_iterator into the forward_list.
__args	Arguments.

Returns

An iterator that points to the inserted data.

This function will insert an object of type `T` constructed with `T(std::forward<Args>(args)...)` after the specified location. Due to the nature of a forward_list this operation can be done in constant time, and does not invalidate iterators and references.

emplace_front()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
template<typename... _Args>
reference std::forward_list< _Tp, _Alloc >::emplace_front (
            _Args &&... __args)  [inline]
```

Constructs object in forward_list at the front of the list.

Parameters

__args	Arguments.

This function will insert an object of type `Tp` constructed with `Tp(std::forward<Args>(args)...)` at the front of the list Due to the nature of a forward_list this operation can be done in constant time, and does not invalidate iterators and references.

empty()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
bool std::forward_list< _Tp, _Alloc >::empty () const  [inline], [noexcept]
```

Returns true if the forward_list is empty. (Thus begin() would equal end().)
Referenced by insert_after(), and sort().

end() [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list< _Tp, _Alloc >::end () const  [inline], [noexcept]
```

Returns a read-only iterator that points one past the last element in the forward_list. Iteration is done in ordinary element order.

end() [2/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list< _Tp, _Alloc >::end ()  [inline], [noexcept]
```

Returns a read/write iterator that points one past the last element in the forward_list. Iteration is done in ordinary element order.

Referenced by std::forward_list< _Tp, polymorphic_allocator< _Tp > >::assign(), std::forward_list< _Tp, polymorphic_allocator< _Tp > insert_after(), insert_after(), resize(), and resize().

erase_after() [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list< _Tp, _Alloc >::erase_after (
            const_iterator __pos)  [inline]
```

Removes the element pointed to by the iterator following `pos`.

Parameters

__pos	Iterator pointing before element to be erased.

Returns

An iterator pointing to the element following the one that was erased, or `end()` if no such element exists.

This function will erase the element at the given position and thus shorten the forward_list by one.

Due to the nature of a forward_list this operation can be done in constant time, and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Referenced by std::forward_list< _Tp, polymorphic_allocator< _Tp > >::assign(), std::forward_list< _Tp, polymorphic_allocator< _Tp > resize(), and resize().

erase_after() [2/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list< _Tp, _Alloc >::erase_after (
            const_iterator __pos,
            const_iterator __last)  [inline]
```

Remove a range of elements.

Parameters

__pos	Iterator pointing before the first element to be erased.
__last	Iterator pointing to one past the last element to be erased.

Returns

`__last`

This function will erase the elements in the range `(__pos,__last)` and shorten the forward_list accordingly.

This operation is linear time in the size of the range and only invalidates iterators/references to the element being removed.

The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

front() [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
reference std::forward_list< _Tp, _Alloc >::front ()  [inline]
```

Returns a read/write reference to the data at the first element of the forward_list.
Referenced by std::forward_list< _Tp, polymorphic_allocator< _Tp > >::emplace_front().

front() [2/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_reference std::forward_list< _Tp, _Alloc >::front () const  [inline]
```

Returns a read-only (constant) reference to the data at the first element of the forward_list.

get_allocator()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
allocator_type std::forward_list< _Tp, _Alloc >::get_allocator () const  [inline], [noexcept]
```

Get a copy of the memory allocation object.
Referenced by insert_after(), insert_after(), and remove().

insert_after() [1/4]

```
template<typename _Tp, typename _Alloc>
template<typename _InputIterator, typename>
forward_list< _Tp, _Alloc >::iterator forward_list::insert_after (
            const_iterator __pos,
            _InputIterator __first,
            _InputIterator __last)
```

Inserts a range into the forward_list.

Parameters

__pos	An iterator into the forward_list.
__first	An input iterator.
__last	An input iterator.

Returns

An iterator pointing to the last inserted element or __pos if __first == __last.

This function will insert copies of the data in the range [__first, __last) into the forward_list after the location specified by `__pos.
This operation is linear in the number of elements inserted and does not invalidate iterators and references.
References forward_list(), before_begin(), empty(), end(), and get_allocator().

insert_after() [2/4]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list< _Tp, _Alloc >::insert_after (
            const_iterator __pos,
            const _Tp & __val)  [inline]
```

Inserts given value into forward_list after specified iterator.

Parameters

__pos	An iterator into the forward_list.
__val	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value after the specified location. Due to the nature of a forward_list this operation can be done in constant time, and does not invalidate iterators and references.
Referenced by std::forward_list< _Tp, polymorphic_allocator< _Tp > >::assign(), std::forward_list< _Tp, polymorphic_allocator< _Tp > std::forward_list< _Tp, polymorphic_allocator< _Tp > >::insert_after(), and resize().

insert_after() [3/4]

```
template<typename _Tp, typename _Alloc>
forward_list< _Tp, _Alloc >::iterator forward_list::insert_after (
            const_iterator __pos,
            size_type __n,
            const _Tp & __val)
```

Inserts a number of copies of given data into the forward_list.

Parameters

__pos	An iterator into the forward_list.
__n	Number of elements to be inserted.
__val	Data to be inserted.

Returns

An iterator pointing to the last inserted copy of `val` or `pos` if `n == 0`.

This function will insert a specified number of copies of the given data after the location specified by `pos`.
This operation is linear in the number of elements inserted and does not invalidate iterators and references.
References forward_list(), before_begin(), end(), and get_allocator().

insert_after() [4/4]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list< _Tp, _Alloc >::insert_after (
            const_iterator __pos,
            std::initializer_list< _Tp > __il)  [inline]
```

Inserts the contents of an initializer_list into forward_list after the specified iterator.

Parameters

__pos	An iterator into the forward_list.
__il	An initializer_list of value_type.

Returns

An iterator pointing to the last inserted element or `__pos` if `__il` is empty.

This function will insert copies of the data in the initializer_list `__il` into the forward_list before the location specified by `__pos`.
This operation is linear in the number of elements inserted and does not invalidate iterators and references.

max_size()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
size_type std::forward_list< _Tp, _Alloc >::max_size () const  [inline], [noexcept]
```

Returns the largest possible number of elements of forward_list.

merge() [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::merge (
            forward_list< _Tp, _Alloc > && __list)  [inline]
```

Merge sorted lists.

Parameters

__list	Sorted list to merge.

Assumes that both __list and this list are sorted according to operator<(). Merges elements of __list into this list in sorted order, leaving __list empty when complete. Elements in this list precede elements in __list that are equal.
Referenced by std::forward_list< _Tp, polymorphic_allocator< _Tp > >::merge().

merge() [2/2]

```
template<typename _Tp, typename _Alloc>
template<typename _Comp>
void forward_list::merge (
            forward_list< _Tp, _Alloc > && __list,
            _Comp __comp)
```

Merge sorted lists according to comparison function.

Parameters

__list	Sorted list to merge.
__comp	Comparison function defining sort order.

Assumes that both __list and this list are sorted according to comp. Merges elements of __list into this list in sorted order, leaving __list empty when complete. Elements in this list precede elements in __list that are equivalent according to comp().
References forward_list(), std::__addressof(), and std::move().

operator=() [1/3]

```
template<typename _Tp, typename _Alloc>
forward_list< _Tp, _Alloc > & forward_list::operator= (
            const forward_list< _Tp, _Alloc > & __list)
```

The forward_list assignment operator.

Parameters

__list	A forward_list of identical element and allocator types.

All the elements of __list are copied.
Whether the allocator is copied depends on the allocator traits.
References forward_list(), std::__addressof(), assign(), and clear().

operator=() [2/3]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
forward_list & std::forward_list< _Tp, _Alloc >::operator= (
            forward_list< _Tp, _Alloc > && __list)  [inline], [noexcept]
```

The forward_list move assignment operator.

Parameters

__list	A forward_list of identical element and allocator types.

The contents of `__list` are moved into this forward_list (without copying, if the allocators permit it).
Afterwards *__list* is a valid, but unspecified forward_list
Whether the allocator is moved depends on the allocator traits.

operator=() [3/3]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
forward_list & std::forward_list< _Tp, _Alloc >::operator= (
            std::initializer_list< _Tp > __il)  [inline]
```

The forward_list initializer list assignment operator.

Parameters

_↩ _il	An initializer_list of value_type.

Replace the contents of the forward_list with copies of the elements in the initializer_list `__il`. This is linear in `__↩ il.size()`.

pop_front()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::pop_front ()  [inline]
```

Removes first element.
This is a typical stack operation. It shrinks the forward_list by one. Due to the nature of a forward_list this operation can be done in constant time, and only invalidates iterators/references to the element being removed.
Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop_front()` is called.

push_front()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::push_front (
            const _Tp & __val)  [inline]
```

Add data to the front of the forward_list.

Parameters

__val	Data to be added.

This is a typical stack operation. The function creates an element at the front of the forward_list and assigns the given data to it. Due to the nature of a forward_list this operation can be done in constant time, and does not invalidate iterators and references.

remove()

```
template<typename _Tp, typename _Alloc>
auto forward_list::remove (
            const _Tp & __val)
```

Remove all elements equal to value.

Parameters

__val	The value to remove.

Removes every element in the list equal to `__val`. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.
References forward_list(), cbefore_begin(), get_allocator(), and splice_after().

remove_if()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
template<typename _Pred>
__remove_return_type std::forward_list< _Tp, _Alloc >::remove_if (
            _Pred __pred)
```

Remove all elements satisfying a predicate.

Parameters

__pred	Unary predicate function or object.

Removes every element in the list for which the predicate returns true. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

resize() [1/2]

```
template<typename _Tp, typename _Alloc>
void forward_list::resize (
            size_type __sz)
```

Resizes the forward_list to the specified number of elements.

Parameters

__sz	Number of elements the forward_list should contain.

This function will resize the forward_list to the specified number of elements. If the number is smaller than the forward_list's current number of elements the forward_list is truncated, otherwise the forward_list is extended and the new elements are default constructed.
References forward_list(), before_begin(), end(), and erase_after().

resize() [2/2]

```
template<typename _Tp, typename _Alloc>
void forward_list::resize (
            size_type __sz,
            const value_type & __val)
```

Resizes the forward_list to the specified number of elements.

Parameters

__sz	Number of elements the forward_list should contain.
__val	Data with which new elements should be populated.

This function will resize the forward_list to the specified number of elements. If the number is smaller than the forward_list's current number of elements the forward_list is truncated, otherwise the forward_list is extended and new elements are populated with given data.
References forward_list(), before_begin(), end(), erase_after(), and insert_after().

reverse()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::reverse ()  [inline], [noexcept]
```

Reverse the elements in list.
Reverse the order of elements in the list in linear time.

sort() [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::sort ()  [inline]
```

Sort the elements of the list.
Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.
Referenced by std::forward_list< _Tp, polymorphic_allocator< _Tp > >::sort().

sort() [2/2]

```
template<typename _Tp, class _Alloc>
template<typename _Comp>
void forward_list::sort (
            _Comp __comp)
```

Sort the forward_list using a comparison function.
Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.
References forward_list(), and empty().

splice_after() [1/4]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::splice_after (
            const_iterator __pos,
            forward_list< _Tp, _Alloc > && ,
            const_iterator __before,
            const_iterator __last)  [inline], [noexcept]
```

Insert range from another forward_list.

Parameters

__pos	Iterator referencing the element to insert after.
__list	Source list.
__before	Iterator referencing before the start of range in `__list`.
__last	Iterator referencing the end of range in `__list`.

Removes elements in the range `(__before,__last)` and inserts them after `__pos` in constant time.
Undefined if `__pos` is in `(__before,__last)`.

splice_after() [2/4]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::splice_after (
            const_iterator __pos,
            forward_list< _Tp, _Alloc > && __list)  [inline], [noexcept]
```

Insert contents of another forward_list.

Parameters

__pos	Iterator referencing the element to insert after.
__list	Source list.

The elements of `list` are inserted in constant time after the element referenced by `pos`. `list` becomes an empty list.
Requires `this != &x`.
Referenced by remove(), and std::forward_list< _Tp, polymorphic_allocator< _Tp > >::splice_after().

splice_after() [3/4]

```
template<typename _Tp, typename _Alloc>
void forward_list::splice_after (
            const_iterator __pos,
            forward_list< _Tp, _Alloc > && __list,
            const_iterator __i)  [noexcept]
```

Insert element from another forward_list.

Parameters

__pos	Iterator referencing the element to insert after.
__list	Source list.
__i	Iterator referencing the element before the element to move.

Removes the element in list `__list` referenced by `__i` and inserts it into the current list after `__pos`.
References forward_list().

splice_after() [4/4]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::splice_after (
            const_iterator __pos,
            forward_list< _Tp, _Alloc > & ,
            const_iterator __before,
            const_iterator __last)  [inline], [noexcept]
```

Insert range from another forward_list.

Parameters

__pos	Iterator referencing the element to insert after.
__list	Source list.
__before	Iterator referencing before the start of range in `__list`.
__last	Iterator referencing the end of range in `__list`.

Removes elements in the range `(__before,__last)` and inserts them after `__pos` in constant time.
Undefined if `__pos` is in `(__before,__last)`.

swap()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::swap (
            forward_list< _Tp, _Alloc > & __list)  [inline], [noexcept]
```

Swaps data with another forward_list.

Parameters

__list	A forward_list of the same element and allocator types.

This exchanges the elements between two lists in constant time. Note that the global `std::swap()` function is overloaded such that `std::swap(l1, l2)` will feed to this function.
Whether the allocators are swapped depends on the allocator traits.

unique() [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
__remove_return_type std::forward_list< _Tp, _Alloc >::unique ()  [inline]
```

Remove consecutive duplicate elements.
For each consecutive set of elements with the same value, remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.
Referenced by std::forward_list< _Tp, polymorphic_allocator< _Tp > >::unique().

unique() [2/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
template<typename _BinPred>
__remove_return_type std::forward_list< _Tp, _Alloc >::unique (
            _BinPred __binary_pred)
```

Remove consecutive elements satisfying a predicate.

Parameters

__binary_pred	Binary predicate function or object.

For each consecutive set of elements [first,last) that satisfy predicate(first,i) where i is an iterator in [first,last), remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.
The documentation for this class was generated from the following files:

- forward_list.h
- forward_list.tcc

5.434 std::fpos< _StateT > Class Template Reference

```
#include <postypes.h>
```

Public Member Functions

- **fpos** (const fpos &)=default
- fpos (streamoff __off)
- operator streamoff () const

- fpos operator+ (streamoff __off) const
- fpos & operator+= (streamoff __off)
- streamoff operator- (const fpos &__other) const
- fpos operator- (streamoff __off) const
- fpos & operator-= (streamoff __off)
- fpos & **operator=** (const fpos &)=default
- _StateT state () const
- void state (_StateT __st)

5.434.1 Detailed Description

template<typename _StateT>
class std::fpos< _StateT >

Class representing stream positions.
The standard places no requirements upon the template parameter StateT. In this implementation StateT must be DefaultConstructible, CopyConstructible and Assignable. The standard only requires that fpos should contain a member of type StateT. In this implementation it also contains an offset stored as a signed integer.

Parameters

StateT	Type passed to and returned from state().

5.434.2 Constructor & Destructor Documentation

fpos()

```
template<typename _StateT>
std::fpos< _StateT >::fpos (
            streamoff __off)  [inline]
```

Construct position from offset.

5.434.3 Member Function Documentation

operator streamoff()

```
template<typename _StateT>
std::fpos< _StateT >::operator streamoff () const  [inline]
```

Convert to streamoff.

operator+()

```
template<typename _StateT>
fpos std::fpos< _StateT >::operator+ (
            streamoff __off) const  [inline]
```

Add position and offset.

operator+=()

```
template<typename _StateT>
fpos & std::fpos< _StateT >::operator+= (
            streamoff __off)  [inline]
```

Add offset to this position.

operator-() [1/2]

```
template<typename _StateT>
streamoff std::fpos< _StateT >::operator- (
            const fpos< _StateT > & __other) const  [inline]
```

Subtract position to return offset.

operator-() [2/2]

```
template<typename _StateT>
fpos std::fpos< _StateT >::operator- (
            streamoff __off) const  [inline]
```

Subtract offset from position.

operator-=()

```
template<typename _StateT>
fpos & std::fpos< _StateT >::operator-= (
            streamoff __off)  [inline]
```

Subtract offset from this position.

state() [1/2]

```
template<typename _StateT>
_StateT std::fpos< _StateT >::state () const  [inline]
```

Return the last set value of *st*.

state() [2/2]

```
template<typename _StateT>
void std::fpos< _StateT >::state (
            _StateT __st)  [inline]
```

Remember the value of *st*.

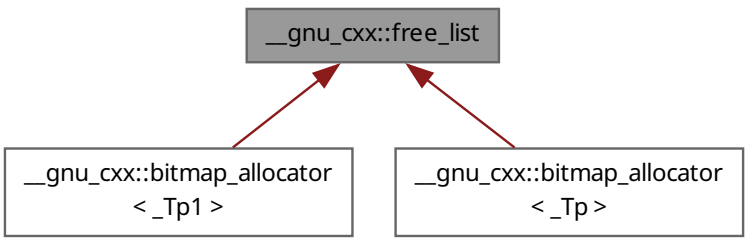
The documentation for this class was generated from the following file:

- postypes.h

5.435 __gnu_cxx::free_list Class Reference

```
#include <bitmap_allocator.h>
```

Inheritance diagram for __gnu_cxx::free_list:



Public Types

- typedef __mutex **__mutex_type**
- typedef vector_type::iterator **iterator**
- typedef std::size_t * **value_type**
- typedef __detail::__mini_vector< value_type > **vector_type**

Public Member Functions

- void _M_clear ()
- std::size_t * _M_get (std::size_t __sz)
- void _M_insert (std::size_t *__addr) throw ()

5.435.1 Detailed Description

The free list class for managing chunks of memory to be given to and returned by the bitmap_allocator.

5.435.2 Member Function Documentation

_M_clear()

```
void __gnu_cxx::free_list::_M_clear ()
```

This function just clears the internal Free List, and gives back all the memory to the OS.

_M_get()

```
std::size_t * __gnu_cxx::free_list::_M_get (
            std::size_t __sz)
```

This function gets a block of memory of the specified size from the free list.

Parameters

__sz	The size in bytes of the memory required.

Returns

A pointer to the new memory block of size at least equal to that requested.

_M_insert()

```
void __gnu_cxx::free_list::_M_insert (
            std::size_t * __addr) throw ( )   [inline]
```

This function returns the block of memory to the internal free list.

Parameters

__addr	The pointer to the memory block that was given by a call to the _M_get function.

Referenced by __gnu_cxx::bitmap_allocator< _Tp1 >::_M_deallocate_single_object().

The documentation for this class was generated from the following file:

- bitmap_allocator.h

5.436 std::from_chars_result Struct Reference

`#include <charconv>`

Public Attributes

- errc **ec**
- const char * **ptr**

Friends

- bool **operator==** (const from_chars_result &, const from_chars_result &)=default

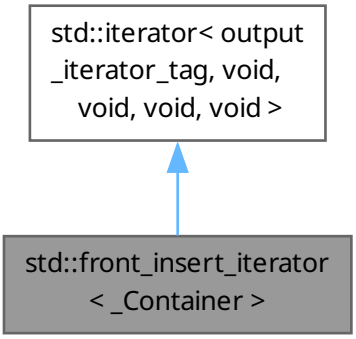
5.436.1 Detailed Description

Result type of std::from_chars.
The documentation for this struct was generated from the following file:

- charconv

5.437 std::front_insert_iterator< _Container > Class Template Reference

`#include <stl_iterator.h>`
Inheritance diagram for std::front_insert_iterator< _Container >:



Public Types

- typedef _Container container_type
- using **difference_type**
- typedef output_iterator_tag iterator_category
- typedef void pointer
- typedef void reference
- typedef void value_type

Public Member Functions

- constexpr front_insert_iterator (_Container &__x)
- constexpr front_insert_iterator & operator* ()
- constexpr front_insert_iterator & operator++ ()
- constexpr front_insert_iterator operator++ (int)
- constexpr front_insert_iterator & operator= (const typename _Container::value_type &__value)
- constexpr front_insert_iterator & **operator=** (typename _Container::value_type &&__value)

Protected Attributes

- _Container * **container**

5.437.1 Detailed Description

template<typename _Container>
class std::front_insert_iterator< _Container >

Turns assignment into insertion.
These are output iterators, constructed from a container-of-T. Assigning a T to the iterator prepends it to the container using push_front.
Tip: Using the front_inserter function to create these iterators can save typing.

5.437.2 Member Typedef Documentation

container_type

```
template<typename _Container>
typedef _Container std::front_insert_iterator< _Container >::container_type
```

A nested typedef for the type of whatever container you used.

iterator_category

```
typedef output_iterator_tag std::iterator< output_iterator_tag, void, void, void, void >::iterator↩
_category  [inherited]
```

One of the tag types.

pointer

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::pointer  [inherited]
```

This type represents a pointer-to-value_type.

reference

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::reference  [inherited]
```

This type represents a reference-to-value_type.

value_type

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::value_type  [inherited]
```

The type "pointed to" by the iterator.

5.437.3 Constructor & Destructor Documentation

front_insert_iterator()

```
template<typename _Container>
std::front_insert_iterator< _Container >::front_insert_iterator (
            _Container & __x)  [inline], [explicit], [constexpr]
```

The only way to create this iterator is with a container.
References std::__addressof().

5.437.4 Member Function Documentation

operator*()

```
template<typename _Container>
front_insert_iterator & std::front_insert_iterator< _Container >::operator* ()  [inline], [nodiscard],
[constexpr]
```

Simply returns *this.

operator++() [1/2]

```
template<typename _Container>
front_insert_iterator & std::front_insert_iterator< _Container >::operator++ ()  [inline], [constexpr]
```

Simply returns *this. (This iterator does not *move*.)

operator++() [2/2]

```
template<typename _Container>
front_insert_iterator std::front_insert_iterator< _Container >::operator++ (
            int )  [inline], [constexpr]
```

Simply returns *this. (This iterator does not *move*.)

operator=()

```
template<typename _Container>
front_insert_iterator & std::front_insert_iterator< _Container >::operator= (
            const typename _Container::value_type & __value)  [inline], [constexpr]
```

Parameters

__value	An instance of whatever type container_type::const_reference is; presumably a reference-to-const T for container<T>.

Returns

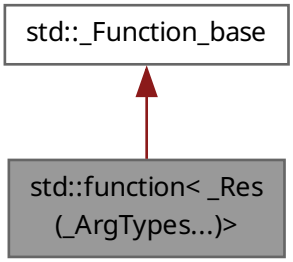
This iterator, for chained operations.

This kind of iterator doesn't really have a *position* in the container (you can think of the position as being permanently at the front, if you like). Assigning a value to the iterator will always prepend the value to the front of the container.
The documentation for this class was generated from the following file:

- bits/stl_iterator.h

5.438 std::function< _Res(_ArgTypes...)> Class Template Reference

`#include <std_function.h>`
Inheritance diagram for std::function< _Res(_ArgTypes...)>:



Public Types

- typedef _Res **result_type**

Public Member Functions

- function () noexcept
- template<typename _Functor, typename _Constraints = _Requires<_Callable<_Functor>>>
  function (_Functor &&__f) noexcept(_Handler< _Functor >::template _S_nothrow_init< _Functor >())
- function (const function &__x)
- function (function &&__x) noexcept
- function (nullptr_t) noexcept
- operator bool () const noexcept
- _Res operator() (_ArgTypes... __args) const
- template<typename _Functor>
  _Requires< _Callable< _Functor >, function & > operator= (_Functor &&__f) noexcept(_Handler< _Functor >::template _S_nothrow_init< _Functor >())
- function & operator= (const function &__x)
- function & operator= (function &&__x) noexcept
- function & operator= (nullptr_t) noexcept
- template<typename _Functor>
  function & operator= (reference_wrapper< _Functor > __f) noexcept
- void swap (function &__x) noexcept
- const type_info & target_type () const noexcept

- template<typename _Functor>
  _Functor * target () noexcept
- template<typename _Functor>
  const _Functor * target () const noexcept

5.438.1 Detailed Description

template<typename _Res, typename... _ArgTypes>
class std::function< _Res(_ArgTypes...)>

Polymorphic function wrapper.

Since

C++11

5.438.2 Constructor & Destructor Documentation

function() [1/5]

```
template<typename _Res, typename... _ArgTypes>
std::function< _Res(_ArgTypes...)>::function () [inline], [noexcept]
```

Default construct creates an empty function call wrapper.

Postcondition

`!(bool)*this`

References function().
Referenced by function(), function(), function(), function(), function(), operator=(), operator=(), operator=(), operator=(), operator=(), swap(), and target().

function() [2/5]

```
template<typename _Res, typename... _ArgTypes>
std::function< _Res(_ArgTypes...)>::function (
            nullptr_t ) [inline], [noexcept]
```

Creates an empty function call wrapper.

Postcondition

!(bool)*this

References function().

function() [3/5]

```
template<typename _Res, typename... _ArgTypes>
std::function< _Res(_ArgTypes...)>::function (
            const function< _Res(_ArgTypes...)> & __x) [inline]
```

Function copy constructor.

Parameters

_↩ _x	A function object with identical call signature.

Postcondition

`bool(*this) == bool(__x)`

The newly-created function contains a copy of the target of `__x` (if it has one).
References function().

function() [4/5]

```
template<typename _Res, typename... _ArgTypes>
std::function< _Res(_ArgTypes...)>::function (
            function< _Res(_ArgTypes...)> && __x) [inline], [noexcept]
```

Function move constructor.

Parameters

_↩ _x	A function object rvalue with identical call signature.

The newly-created function contains the target of __x (if it has one).
References function().

function() [5/5]

```
template<typename _Res, typename... _ArgTypes>
template<typename _Functor, typename _Constraints = _Requires<_Callable<_Functor>>>
std::function< _Res(_ArgTypes...)>::function (
            _Functor && __f) [inline], [noexcept]
```

Builds a function that targets a copy of the incoming function object.

Parameters

↩ _↩ ↩ _↩ f	A function object that is callable with parameters of type `ArgTypes...` and returns a value convertible to `Res`.

The newly-created function object will target a copy of __f. If __f is `reference_wrapper<F>`, then this function object will contain a reference to the function object __f.get(). If __f is a null function pointer, null pointer-to-member, or empty `std::function`, the newly-created object will be empty.
If __f is a non-null function pointer or an object of type `reference_wrapper<F>`, this function will not throw.
References function(), and std::forward().

5.438.3 Member Function Documentation

operator bool()

```
template<typename _Res, typename... _ArgTypes>
std::function< _Res(_ArgTypes...)>::operator bool () const [inline], [explicit], [noexcept]
```

Determine if the function wrapper has a target.

Returns

`true` when this function object contains a target, or `false` when it is empty.

This function will not throw exceptions.

operator()()

```
template<typename _Res, typename... _ArgTypes>
_Res std::function< _Res(_ArgTypes...)>::operator() (
            _ArgTypes... __args) const [inline]
```

Invokes the function targeted by `*this`.

Returns

the result of the target.

Exceptions

`bad_function_call`	when `!(bool)*this`

The function call operator invokes the target function object stored by `this`.
References std::forward(), and operator()().
Referenced by operator()().

operator=() [1/5]

```
template<typename _Res, typename...  _ArgTypes>
template<typename _Functor>
_Requires< _Callable< _Functor >, function & > std::function< _Res(_ArgTypes...)>::operator= (
            _Functor && __f)  [inline], [noexcept]
```

Function assignment to a new target.

Parameters

↩ _↩ ↩ _↩ *f*	A function object that is callable with parameters of type `_ArgTypes...` and returns a value convertible to `_Res`.

Returns

`*this`

Since

C++11

This function object wrapper will target a copy of `__f`. If `__f` is `reference_wrapper<F>`, then this function object will contain a reference to the function object `__f.get()`. If `__f` is a null function pointer or null pointer-to-member, this object will be empty.
If `__f` is a non-null function pointer or an object of type `reference_wrapper<F>`, this function will not throw.
References function(), std::forward(), and operator=().

operator=() [2/5]

```
template<typename _Res, typename...  _ArgTypes>
function & std::function< _Res(_ArgTypes...)>::operator= (
            const function< _Res(_ArgTypes...)> & __x)  [inline]
```

Function assignment operator.

Parameters

_↩ *_x*	A function with identical call signature.

Postcondition

```
(bool)*this == (bool)x
```

Returns

```
*this
```

The target of __x is copied to *this. If __x has no target, then *this will be empty.
If __x targets a function pointer or a reference to a function object, then this operation will not throw an exception.
References function(), and operator=().
Referenced by operator=(), operator=(), operator=(), operator=(), and operator=().

operator=() [3/5]

```
template<typename _Res, typename...  _ArgTypes>
function & std::function< _Res(_ArgTypes...)>::operator= (
            function< _Res(_ArgTypes...)> && __x)  [inline], [noexcept]
```

Function move-assignment operator.

Parameters

_↩ _x	A function rvalue with identical call signature.

Returns

```
*this
```

The target of __x is moved to *this. If __x has no target, then *this will be empty.
If __x targets a function pointer or a reference to a function object, then this operation will not throw an exception.
References function(), std::move(), and operator=().

operator=() [4/5]

```
template<typename _Res, typename...  _ArgTypes>
function & std::function< _Res(_ArgTypes...)>::operator= (
            nullptr_t )  [inline], [noexcept]
```

Function assignment to empty.

Postcondition

```
!(bool)*this
```

Returns

```
*this
```

The target of *this is deallocated, leaving it empty.
References function(), and operator=().

operator=() [5/5]

```
template<typename _Res, typename...  _ArgTypes>
template<typename _Functor>
function & std::function< _Res(_ArgTypes...)>::operator= (
            reference_wrapper< _Functor > __f)  [inline], [noexcept]
```

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.
References function(), and operator=().

swap()

```
template<typename _Res, typename... _ArgTypes>
void std::function< _Res(_ArgTypes...)>::swap (
            function< _Res(_ArgTypes...)> & __x) [inline], [noexcept]
```

Swap the targets of two function objects.

Parameters

_↩ _x	A function with identical call signature.

Swap the targets of `this` function object and `__f`. This function will not throw exceptions.
References function(), and swap().
Referenced by swap().

target() [1/2]

```
template<typename _Res, typename... _ArgTypes>
template<typename _Functor>
const _Functor * std::function< _Res(_ArgTypes...)>::target () const [inline], [noexcept]
```

Access the stored target function object.

Returns

Returns a pointer to the stored target function object, if `typeid(_Functor).equals(target_type());` otherwise, a null pointer.

This function does not throw exceptions.
References target(), and target_type().

target() [2/2]

```
template<typename _Res, typename... _ArgTypes>
template<typename _Functor>
_Functor * std::function< _Res(_ArgTypes...)>::target () [inline], [noexcept]
```

Access the stored target function object.

Returns

Returns a pointer to the stored target function object, if `typeid(_Functor).equals(target_type());` otherwise, a null pointer.

This function does not throw exceptions.
References function(), and target().
Referenced by target(), and target().

target_type()

```
template<typename _Res, typename... _ArgTypes>
const type_info & std::function< _Res(_ArgTypes...)>::target_type () const [inline], [noexcept]
```

Determine the type of the target of this function object wrapper.

Returns

the type identifier of the target function object, or `typeid(void)` if `!(bool)*this`.

This function will not throw exceptions.
References target_type().
Referenced by target(), and target_type().
The documentation for this class was generated from the following file:

- std_function.h

5.439 std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> Class Template Reference

`#include <functional>`

Public Member Functions

- template<typename _Fn, typename _Vt = remove_reference_t<_Fn>>
  requires (!is_same_v<remove_cv_t<_Vt>, function_ref>) && (!is_member_pointer_v<_Vt>) && (!is_function_v<_Vt>) && __is_↩
  invocable_using<_Vt _GLIBCXX_MOF_CV&>
  constexpr function_ref (_Fn &&__f) noexcept
- template<typename _Fn>
  requires is_function_v<_Fn> && __is_invocable_using<_Fn*>
  function_ref (_Fn *__fn) noexcept
- template<auto __fn>
  requires __is_invocable_using<const decltype(__fn)&>
  constexpr function_ref (nontype_t< __fn >) noexcept
- template<auto __fn, typename _Td>
  requires __is_invocable_using<const decltype(__fn)&, _Td _GLIBCXX_MOF_CV*>
  constexpr function_ref (nontype_t< __fn >, _Td _GLIBCXX_MOF_CV *__ptr) noexcept
- template<auto __fn, typename _Up, typename _Td = remove_reference_t<_Up>>
  requires (!is_rvalue_reference_v<_Up&&>) && __is_invocable_using<const decltype(__fn)&, _Td _GLIBCXX_MOF_CV&>
  constexpr function_ref (nontype_t< __fn >, _Up &&__ref) noexcept
- _Res operator() (_ArgTypes... __args) const noexcept(_Noex)
- template<typename _Tp>
  requires (!is_same_v<_Tp, function_ref>) && (!is_pointer_v<_Tp>) && (!__is_nontype_v<_Tp>)
  function_ref & **operator=** (_Tp)=delete

5.439.1 Detailed Description

template<typename _Res, typename... _ArgTypes, bool _Noex>
class std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>

Non-owning polymorphic function wrapper.

Since

C++26

The `std::function_ref` class template is a non-owning call wrapper, that refers to a bound object. Using function_ref outside of the lifetime of the bound object has undefined behavior.
It supports const-qualification and no-throw guarantees. The qualifications and exception-specification of the signature are respected when invoking the reference function.

5.439.2 Constructor & Destructor Documentation

function_ref() [1/5]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<typename _Fn>
requires is_function_v<_Fn> && __is_invocable_using<_Fn*>
std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::function_ref (
            _Fn * __fn) [inline], [noexcept]
```

Target and bound object is function pointed by parameter.

function_ref() [2/5]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<typename _Fn, typename _Vt = remove_reference_t<_Fn>>
requires (!is_same_v<remove_cv_t<_Vt>, function_ref>) && (!is_member_pointer_v<_Vt>) && (!is_↩
function_v<_Vt>) && __is_invocable_using<_Vt _GLIBCXX_MOF_CV&>
std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::function_ref (
            _Fn && __f) [inline], [constexpr], [noexcept]
```

Target and bound object is object referenced by parameter.
References std::addressof().

function_ref() [3/5]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<auto __fn>
requires __is_invocable_using<const decltype(__fn)&>
std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::function_ref (
            nontype_t< __fn > ) [inline], [constexpr], [noexcept]
```

Target object is __fn. There is no bound object.

function_ref() [4/5]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<auto __fn, typename _Up, typename _Td = remove_reference_t<_Up>>
requires (!is_rvalue_reference_v<_Up&&>) && __is_invocable_using<const decltype(__fn)&, _Td _↩
GLIBCXX_MOF_CV&>
std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::function_ref (
            nontype_t< __fn > ,
            _Up && __ref) [inline], [constexpr], [noexcept]
```

Target object is equivalent to std::bind_front<_fn>(std::ref(__ref)). Bound object is object referenced by second parameter.
References std::addressof().

function_ref() [5/5]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<auto __fn, typename _Td>
requires __is_invocable_using<const decltype(__fn)&, _Td _GLIBCXX_MOF_CV*>
std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::function_ref (
            nontype_t< __fn > ,
            _Td _GLIBCXX_MOF_CV * __ptr) [inline], [constexpr], [noexcept]
```

Target object is equivalent to std::bind_front<_fn>(__ptr). Bound object is object pointed by second parameter (if any).

5.439.3 Member Function Documentation

operator()()

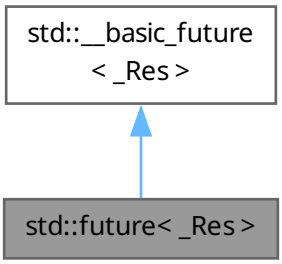
```
template<typename _Res, typename...  _ArgTypes, bool _Noex>
_Res std::function_ref< _Res(_ArgTypes...)  _GLIBCXX_MOF_CV noexcept(_Noex)>::operator() (
            _ArgTypes...  __args) const  [inline], [noexcept]
```

Invoke the target object.
The bound object will be invoked using the supplied arguments, and as const or non-const, as dictated by the template arguments of the `function_ref` specialization.
References std::forward().
The documentation for this class was generated from the following file:

- funcref_impl.h

5.440 std::future< _Res > Class Template Reference

`#include <future>`
Inheritance diagram for std::future< _Res >:



Public Member Functions

- **future** (const future &)=delete
- future (future &&__uf) noexcept
- _Res get ()
- future & **operator=** (const future &)=delete
- future & **operator=** (future &&__fut) noexcept
- shared_future< _Res > **share** () noexcept
- bool **valid** () const noexcept
- void **wait** () const
- template<typename _Rep, typename _Period>
  future_status **wait_for** (const chrono::duration< _Rep, _Period > &__rel) const
- template<typename _Clock, typename _Duration>
  future_status **wait_until** (const chrono::time_point< _Clock, _Duration > &__abs) const

Protected Types

- typedef __future_base::_Result< _Res > & **__result_type**

Protected Member Functions

- __result_type _M_get_result () const
- void **_M_swap** (__basic_future &__that) noexcept

Friends

- template<typename _Fn, typename... _Args>
  future< __async_result_of< _Fn, _Args... > > **async** (launch, _Fn &&, _Args &&...)
- template<typename>
  class **packaged_task**
- class **promise**< **_Res** >

5.440.1 Detailed Description

template<typename _Res>
class std::future< _Res >

Primary template for future.

5.440.2 Constructor & Destructor Documentation

future()

```
template<typename _Res>
std::future< _Res >::future (
            future< _Res > && __uf)  [inline], [noexcept]
```

Move constructor.

5.440.3 Member Function Documentation

_M_get_result()

```
template<typename _Res>
__result_type std::__basic_future< _Res >::_M_get_result () const  [inline], [protected], [inherited]
```

Wait for the state to be ready and rethrow any stored exception.

get()

```
template<typename _Res>
_Res std::future< _Res >::get ()  [inline]
```

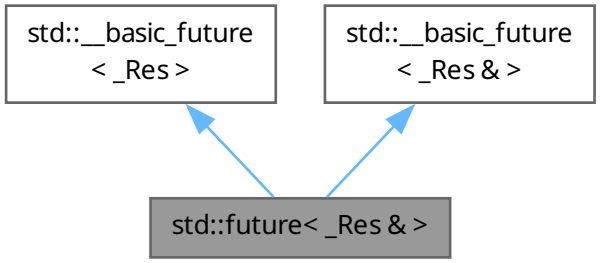
Retrieving the value.
The documentation for this class was generated from the following file:

- future

5.441 std::future< _Res & > Class Template Reference

```
#include <future>
```

Inheritance diagram for std::future< _Res & >:



Public Member Functions

- **future** (const future &)=delete
- **future** (const future &)=delete
- future (future &&__uf) noexcept
- future (future &&__uf) noexcept
- _Res get ()
- _Res & get ()
- future & **operator=** (const future &)=delete
- future & **operator=** (const future &)=delete
- future & **operator=** (future &&__fut) noexcept
- future & **operator=** (future &&__fut) noexcept
- shared_future< _Res > **share** () noexcept
- shared_future< _Res & > **share** () noexcept
- bool **valid** () const noexcept
- void **wait** () const
- future_status **wait_for** (const chrono::duration< _Rep, _Period > &__rel) const
- future_status **wait_until** (const chrono::time_point< _Clock, _Duration > &__abs) const

Protected Types

- typedef __future_base::_Result< _Res > & **__result_type**

Protected Member Functions

- __result_type _M_get_result () const
- void **_M_swap** (__basic_future &__that) noexcept

Friends

- future< __async_result_of< _Fn, _Args... > > **async** (launch, _Fn &&, _Args &&...)
- template<typename _Fn, typename... _Args>
  future< __async_result_of< _Fn, _Args... > > **async** (launch, _Fn &&, _Args &&...)
- class **packaged_task**

- template<typename>
  class **packaged_task**
- class **promise< _Res & >**
- class **promise< _Res >**

5.441.1 Detailed Description

template<typename _Res>
class std::future< _Res & >

Partial specialization for future<R&>

5.441.2 Constructor & Destructor Documentation

future() [1/2]

```
template<typename _Res>
std::future< _Res & >::future (
            future< _Res & > && __uf)  [inline], [noexcept]
```

Move constructor.

future() [2/2]

```
std::future< _Res >::future (
            future< _Res & > && __uf)  [inline], [noexcept]
```

Move constructor.

5.441.3 Member Function Documentation

_M_get_result()

```
__result_type std::__basic_future< _Res >::_M_get_result () const  [inline], [protected]
```

Wait for the state to be ready and rethrow any stored exception.

get() [1/2]

```
_Res std::future< _Res >::get ()  [inline]
```

Retrieving the value.

get() [2/2]

```
template<typename _Res>
_Res & std::future< _Res & >::get ()  [inline]
```

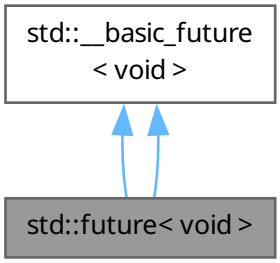
Retrieving the value.
The documentation for this class was generated from the following file:

- future

5.442 std::future< void > Class Reference

```
#include <future>
```

Inheritance diagram for std::future< void >:



Public Member Functions

- **future** (const future &)=delete
- **future** (const future &)=delete
- future (future &&__uf) noexcept
- future (future &&__uf) noexcept
- void get ()
- void get ()
- future & **operator=** (const future &)=delete
- future & **operator=** (const future &)=delete
- future & **operator=** (future &&__fut) noexcept
- future & **operator=** (future &&__fut) noexcept
- shared_future< void > **share** () noexcept
- shared_future< void > **share** () noexcept
- bool **valid** () const noexcept
- void **wait** () const
- future_status **wait_for** (const chrono::duration< _Rep, _Period > &__rel) const
- future_status **wait_until** (const chrono::time_point< _Clock, _Duration > &__abs) const

Protected Types

- typedef __future_base::_Result< void > & **__result_type**

Protected Member Functions

- __result_type _M_get_result () const
- void **_M_swap** (__basic_future &__that) noexcept

Friends

- future< __async_result_of< _Fn, _Args... > > **async** (launch, _Fn &&, _Args &&...)
- template<typename _Fn, typename... _Args>
  future< __async_result_of< _Fn, _Args... > > **async** (launch, _Fn &&, _Args &&...)
- class **packaged_task**

- template<typename>
  class **packaged_task**
- class **promise**< **_Res** >
- class **promise**< **void** >

5.442.1 Detailed Description

Explicit specialization for future<void>

5.442.2 Constructor & Destructor Documentation

future() [1/2]

```
std::future< void >::future (
            future< void > && __uf)  [inline], [noexcept]
```

Move constructor.

future() [2/2]

```
std::future< void >::future (
            future< void > && __uf)  [inline], [noexcept]
```

Move constructor.

5.442.3 Member Function Documentation

_M_get_result()

```
__result_type std::__basic_future< void >::_M_get_result () const  [inline], [protected]
```

Wait for the state to be ready and rethrow any stored exception.

get() [1/2]

```
void std::future< void >::get ()  [inline]
```

Retrieving the value.

get() [2/2]

```
void std::future< void >::get ()  [inline]
```

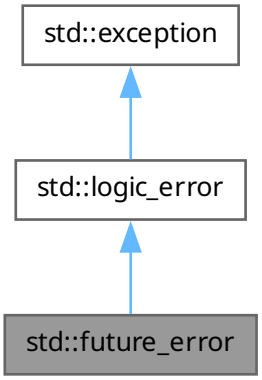
Retrieving the value.
The documentation for this class was generated from the following file:

- future

5.443 std::future_error Class Reference

```
#include <future>
```

Inheritance diagram for std::future_error:



Public Member Functions

- **future_error** (future_errc __errc)
- const error_code & **code** () const noexcept
- virtual const char * what () const noexcept

Friends

- void **__throw_future_error** (int)

5.443.1 Detailed Description

Exception type thrown by futures.

Since

C++11

5.443.2 Member Function Documentation

what()

```
virtual const char * std::future_error::what () const  [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).
Reimplemented from std::logic_error.
The documentation for this class was generated from the following file:

- future

5.444 std::gamma_distribution< _RealType > Class Template Reference

```
#include <random>
```

Classes

- struct param_type

Public Types

- typedef _RealType result_type

Public Member Functions

- gamma_distribution ()
- gamma_distribution (_RealType __alpha_val, _RealType __beta_val=_RealType(1))
- **gamma_distribution** (const param_type &__p)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- _RealType alpha () const
- _RealType beta () const
- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- template<typename _RealType1, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::gamma_distribution< _RealType1 > &__x)
- bool operator== (const gamma_distribution &__d1, const gamma_distribution &__d2)
- template<typename _RealType1, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::gamma_distribution< _RealType1 > &__x)

5.444.1 Detailed Description

template<typename _RealType = double>
class std::gamma_distribution< _RealType >

A gamma continuous distribution for random numbers.
The formula for the gamma probability density function is:

$$p(x|\alpha,\beta) = \frac{1}{\beta\Gamma(\alpha)}(x/\beta)^{\alpha-1}e^{-x/\beta}$$

Since

C++11

5.444.2 Member Typedef Documentation

result_type

```
template<typename _RealType = double>
typedef _RealType std::gamma_distribution< _RealType >::result_type
```

The type of the range of the distribution.

5.444.3 Constructor & Destructor Documentation

gamma_distribution() [1/2]

```
template<typename _RealType = double>
std::gamma_distribution< _RealType >::gamma_distribution ()  [inline]
```

Constructs a gamma distribution with parameters 1 and 1.

gamma_distribution() [2/2]

```
template<typename _RealType = double>
std::gamma_distribution< _RealType >::gamma_distribution (
            _RealType __alpha_val,
            _RealType __beta_val = _RealType(1))  [inline], [explicit]
```

Constructs a gamma distribution with parameters α and β.

5.444.4 Member Function Documentation

alpha()

```
template<typename _RealType = double>
_RealType std::gamma_distribution< _RealType >::alpha () const  [inline]
```

Returns the α of the distribution.

beta()

```
template<typename _RealType = double>
_RealType std::gamma_distribution< _RealType >::beta () const  [inline]
```

Returns the β of the distribution.

max()

```
template<typename _RealType = double>
result_type std::gamma_distribution< _RealType >::max () const  [inline]
```

Returns the least upper bound value of the distribution.

min()

```
template<typename _RealType = double>
result_type std::gamma_distribution< _RealType >::min () const  [inline]
```

Returns the greatest lower bound value of the distribution.

operator()() [1/2]

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::gamma_distribution< _RealType >::operator() (
            _UniformRandomNumberGenerator & __urng)  [inline]
```

Generating functions.
Referenced by operator()().

operator()() [2/2]

```
template<typename _RealType>
template<typename _UniformRandomNumberGenerator>
gamma_distribution< _RealType >::result_type std::gamma_distribution< _RealType >::operator() (
            _UniformRandomNumberGenerator & __urng,
            const param_type & __param)
```

Marsaglia, G. and Tsang, W. W. "A Simple Method for Generating Gamma Variables" ACM Transactions on Mathematical Software, 26, 3, 363-372, 2000.
References std::log(), and std::pow().

param() [1/2]

```
template<typename _RealType = double>
param_type std::gamma_distribution< _RealType >::param () const  [inline]
```

Returns the parameter set of the distribution.

param() [2/2]

```
template<typename _RealType = double>
void std::gamma_distribution< _RealType >::param (
            const param_type & __param)  [inline]
```

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

reset()

```
template<typename _RealType = double>
void std::gamma_distribution< _RealType >::reset ()  [inline]
```

Resets the distribution state.

5.444.5 Friends And Related Symbol Documentation

operator<<

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::gamma_distribution< _RealType1 > & __x)  [friend]
```

Inserts a gamma_distribution random number distribution __x into the output stream __os.

Parameters

__os	An output stream.
__x	A gamma_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

operator==

```
template<typename _RealType = double>
bool operator== (
             const gamma_distribution< _RealType > & __d1,
             const gamma_distribution< _RealType > & __d2)  [friend]
```

Return true if two gamma distributions have the same parameters and the sequences that would be generated are equal.

operator>>

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
             std::basic_istream< _CharT, _Traits > & __is,
             std::gamma_distribution< _RealType1 > & __x)  [friend]
```

Extracts a gamma_distribution random number distribution __x from the input stream __is.

Parameters

_↩ _is	An input stream.
_↩ _x	A gamma_distribution random number generator engine.

Returns

The input stream with __x extracted or in an error state.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

5.445 std::geometric_distribution< _IntType > Class Template Reference

```
#include <random>
```

Classes

- struct param_type

Public Types

- typedef _IntType result_type

Public Member Functions

- **geometric_distribution** (const param_type &__p)
- **geometric_distribution** (double __p)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- double p () const
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- bool operator== (const geometric_distribution &__d1, const geometric_distribution &__d2)

5.445.1 Detailed Description

template<typename _IntType = int>
class std::geometric_distribution< _IntType >

A discrete geometric random number distribution.
The formula for the geometric probability density function is $p(i|p) = p(1-p)^i$ where p is the parameter of the distribution.

Since

C++11

5.445.2 Member Typedef Documentation

result_type

```
template<typename _IntType = int>
typedef _IntType std::geometric_distribution< _IntType >::result_type
```

The type of the range of the distribution.

5.445.3 Member Function Documentation

max()

```
template<typename _IntType = int>
result_type std::geometric_distribution< _IntType >::max () const  [inline]
```

Returns the least upper bound value of the distribution.

min()

```
template<typename _IntType = int>
result_type std::geometric_distribution< _IntType >::min () const  [inline]
```

Returns the greatest lower bound value of the distribution.

operator()()

```
template<typename _IntType = int>
template<typename _UniformRandomNumberGenerator>
result_type std::geometric_distribution< _IntType >::operator() (
            _UniformRandomNumberGenerator & __urng)  [inline]
```

Generating functions.
Referenced by operator()().

p()

```
template<typename _IntType = int>
double std::geometric_distribution< _IntType >::p () const  [inline]
```

Returns the distribution parameter p.

param() [1/2]

```
template<typename _IntType = int>
param_type std::geometric_distribution< _IntType >::param () const  [inline]
```

Returns the parameter set of the distribution.
Referenced by std::operator>>().

param() [2/2]

```
template<typename _IntType = int>
void std::geometric_distribution< _IntType >::param (
            const param_type & __param)  [inline]
```

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

reset()

```
template<typename _IntType = int>
void std::geometric_distribution< _IntType >::reset ()  [inline]
```

Resets the distribution state.
Does nothing for the geometric distribution.

5.445.4 Friends And Related Symbol Documentation

operator==

```
template<typename _IntType = int>
bool operator== (
            const geometric_distribution< _IntType > & __d1,
            const geometric_distribution< _IntType > & __d2)  [friend]
```

Return true if two geometric distributions have the same parameters.

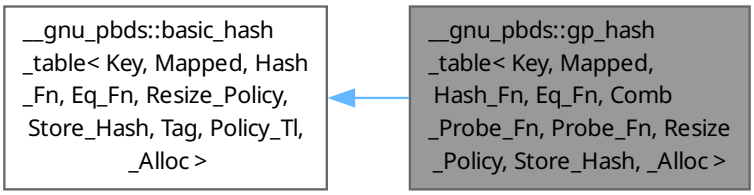
The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

5.446 __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc > Class Template Reference

`#include <assoc_container.hpp>`

Inheritance diagram for __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >:



Public Types

- typedef Comb_Probe_Fn **comb_probe_fn**
- typedef gp_hash_tag **container_category**
- typedef Eq_Fn **eq_fn**
- typedef Hash_Fn **hash_fn**
- typedef Probe_Fn **probe_fn**
- typedef Resize_Policy **resize_policy**

Public Member Functions

- gp_hash_table ()
- **gp_hash_table** (const gp_hash_table &other)
- gp_hash_table (const hash_fn &h)
- gp_hash_table (const hash_fn &h, const eq_fn &e)
- gp_hash_table (const hash_fn &h, const eq_fn &e, const comb_probe_fn &cp)
- gp_hash_table (const hash_fn &h, const eq_fn &e, const comb_probe_fn &cp, const probe_fn &p)
- gp_hash_table (const hash_fn &h, const eq_fn &e, const comb_probe_fn &cp, const probe_fn &p, const resize_policy &rp)
- template<typename It>
  gp_hash_table (It first, It last)
- template<typename It>
  gp_hash_table (It first, It last, const hash_fn &h)
- template<typename It>
  gp_hash_table (It first, It last, const hash_fn &h, const eq_fn &e)
- template<typename It>
  gp_hash_table (It first, It last, const hash_fn &h, const eq_fn &e, const comb_probe_fn &cp)

- template<typename It>
  gp_hash_table (It first, It last, const hash_fn &h, const eq_fn &e, const comb_probe_fn &cp, const probe_fn &p)
- template<typename It>
  gp_hash_table (It first, It last, const hash_fn &h, const eq_fn &e, const comb_probe_fn &cp, const probe_fn &p, const resize_policy &rp)
- gp_hash_table & **operator=** (const gp_hash_table &other)
- void **swap** (gp_hash_table &other)

5.446.1 Detailed Description

template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
class __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >

A general-probing hash-based associative container.

Template Parameters

Key	Key type.
Mapped	Map type.
Hash_Fn	Hashing functor.
Eq_Fn	Equal functor.
Comb_Probe_Fn	Combining probe functor. If Hash_Fn is not null_type, then this is the ranged-probe functor; otherwise, this is the range-hashing functor. XXX See Design::Hash-Based Containers::Hash Policies.
Probe_Fn	Probe functor.
Resize_Policy	Resizes hash.
Store_Hash	Indicates whether the hash value will be stored along with each key. If Hash_Fn is null_type, then the container will not compile if this value is true
_Alloc	Allocator type.

Base tag choices are: gp_hash_tag.
Base is basic_hash_table.

5.446.2 Constructor & Destructor Documentation

gp_hash_table() [1/12]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >::gp_hash_table ()  [inline]
```

Default constructor.

gp_hash_table() [2/12]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↩
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↩
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↩
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
            const hash_fn & h)  [inline]
```

Constructor taking some policy objects. r_hash_fn will be copied by the hash_fn object of the container object.

gp_hash_table() [3/12]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↩
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↩
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↩
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
            const hash_fn & h,
            const eq_fn & e)  [inline]
```

Constructor taking some policy objects. r_hash_fn will be copied by the hash_fn object of the container object, and r_eq_fn will be copied by the eq_fn object of the container object.

gp_hash_table() [4/12]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↩
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↩
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↩
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
            const hash_fn & h,
            const eq_fn & e,
            const comb_probe_fn & cp)  [inline]
```

Constructor taking some policy objects. r_hash_fn will be copied by the hash_fn object of the container object, r_eq_fn will be copied by the eq_fn object of the container object, and r_comb_probe_fn will be copied by the comb_probe_fn object of the container object.

gp_hash_table() [5/12]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↩
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↩
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↩
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
            const hash_fn & h,
            const eq_fn & e,
            const comb_probe_fn & cp,
            const probe_fn & p)  [inline]
```

Constructor taking some policy objects. r_hash_fn will be copied by the hash_fn object of the container object, r_eq_fn will be copied by the eq_fn object of the container object, r_comb_probe_fn will be copied by the comb_probe_fn object of the container object, and r_probe_fn will be copied by the probe_fn object of the container object.

gp_hash_table() [6/12]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↩
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↩
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↩
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
             const hash_fn & h,
             const eq_fn & e,
             const comb_probe_fn & cp,
             const probe_fn & p,
             const resize_policy & rp)  [inline]
```

Constructor taking some policy objects. r_hash_fn will be copied by the hash_fn object of the container object, r_eq_fn will be copied by the eq_fn object of the container object, r_comb_probe_fn will be copied by the comb_probe_fn object of the container object, r_probe_fn will be copied by the probe_fn object of the container object, and r_resize_policy will be copied by the Resize_Policy object of the container object.

gp_hash_table() [7/12]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↩
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↩
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↩
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
             It first,
             It last)  [inline]
```

Constructor taking __iterators to a range of value_types. The value_types between first_it and last_it will be inserted into the container object.

gp_hash_table() [8/12]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↩
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↩
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↩
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
             It first,
             It last,
             const hash_fn & h)  [inline]
```

Constructor taking __iterators to a range of value_types and some policy objects. The value_types between first_it and last_it will be inserted into the container object. r_hash_fn will be copied by the hash_fn object of the container object.

gp_hash_table() [9/12]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↩
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↩
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↩
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
             It first,
             It last,
             const hash_fn & h,
             const eq_fn & e)  [inline]
```

Constructor taking __iterators to a range of value_types and some policy objects. The value_types between first_it and last_it will be inserted into the container object. r_hash_fn will be copied by the hash_fn object of the container object, and r_eq_fn will be copied by the eq_fn object of the container object.

gp_hash_table() [10/12]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↩
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↩
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↩
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
             It first,
             It last,
             const hash_fn & h,
             const eq_fn & e,
             const comb_probe_fn & cp)  [inline]
```

Constructor taking __iterators to a range of value_types and some policy objects. The value_types between first_it and last_it will be inserted into the container object. r_hash_fn will be copied by the hash_fn object of the container object, r_eq_fn will be copied by the eq_fn object of the container object, and r_comb_probe_fn will be copied by the comb_probe_fn object of the container object.

gp_hash_table() [11/12]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↩
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↩
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↩
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
             It first,
             It last,
             const hash_fn & h,
             const eq_fn & e,
             const comb_probe_fn & cp,
             const probe_fn & p)  [inline]
```

Constructor taking __iterators to a range of value_types and some policy objects. The value_types between first_it and

last_it will be inserted into the container object. r_hash_fn will be copied by the hash_fn object of the container object, r_eq_fn will be copied by the eq_fn object of the container object, r_comb_probe_fn will be copied by the comb_probe_fn object of the container object, and r_probe_fn will be copied by the probe_fn object of the container object.

gp_hash_table() [12/12]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↩
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_↩
Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>↩
::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
             It first,
             It last,
             const hash_fn & h,
             const eq_fn & e,
             const comb_probe_fn & cp,
             const probe_fn & p,
             const resize_policy & rp)  [inline]
```

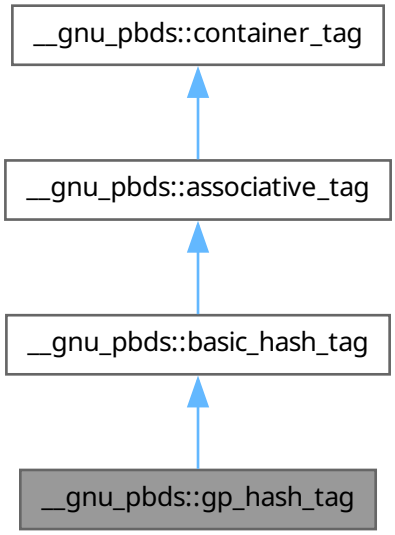
Constructor taking __iterators to a range of value_types and some policy objects. The value_types between first_it and last_it will be inserted into the container object. r_hash_fn will be copied by the hash_fn object of the container object, r_eq_fn will be copied by the eq_fn object of the container object, r_comb_probe_fn will be copied by the comb_↩ probe_fn object of the container object, r_probe_fn will be copied by the probe_fn object of the container object, and r_resize_policy will be copied by the resize_policy object of the container object.
The documentation for this class was generated from the following file:

- assoc_container.hpp

5.447 __gnu_pbds::gp_hash_tag Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for __gnu_pbds::gp_hash_tag:



5.447.1 Detailed Description

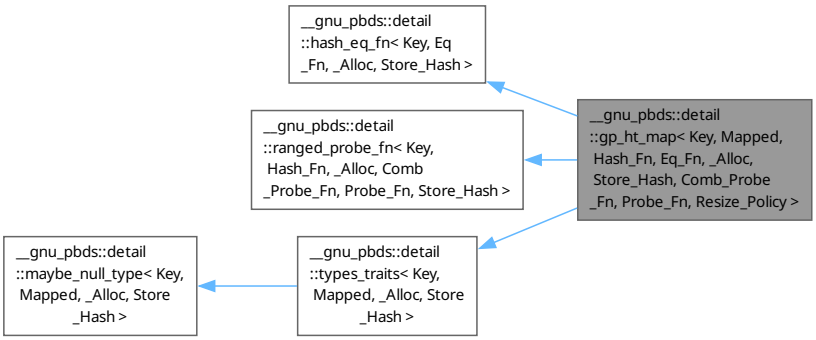
General-probing hash.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

5.448 __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy > Class Template Reference

`#include <gp_ht_map_.hpp>`
Inheritance diagram for __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb↩

_Probe_Fn, Probe_Fn, Resize_Policy >:



Public Types

- enum { **store_hash** }
- typedef _Alloc **allocator_type**
- typedef Comb_Probe_Fn **comb_probe_fn**
- typedef const_iterator_ **const_iterator**
- typedef traits_base::const_pointer **const_pointer**
- typedef traits_base::const_reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef Eq_Fn **eq_fn**
- typedef Hash_Fn **hash_fn**
- typedef iterator_ **iterator**
- typedef traits_base::key_const_pointer **key_const_pointer**
- typedef traits_base::key_const_reference **key_const_reference**
- typedef traits_base::key_pointer **key_pointer**
- typedef traits_base::key_reference **key_reference**
- typedef traits_base::key_type **key_type**
- typedef traits_base::mapped_const_pointer **mapped_const_pointer**
- typedef traits_base::mapped_const_reference **mapped_const_reference**
- typedef traits_base::mapped_pointer **mapped_pointer**
- typedef traits_base::mapped_reference **mapped_reference**
- typedef traits_base::mapped_type **mapped_type**
- typedef __nothrowcopy::indicator **no_throw_indicator**
- typedef point_const_iterator_ **point_const_iterator**
- typedef point_iterator_ **point_iterator**
- typedef traits_base::pointer **pointer**
- typedef Probe_Fn **probe_fn**
- typedef traits_base::reference **reference**
- typedef Resize_Policy **resize_policy**
- typedef _Alloc::size_type **size_type**
- typedef integral_constant< int, Store_Hash > **store_extra**
- typedef stored_data< value_type, size_type, Store_Hash > **stored_data_type**
- typedef traits_base::value_type **value_type**

Public Member Functions

- **gp_ht_map** (const gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy > &)
- **gp_ht_map** (const Hash_Fn &)
- **gp_ht_map** (const Hash_Fn &, const Eq_Fn &)
- **gp_ht_map** (const Hash_Fn &, const Eq_Fn &, const Comb_Probe_Fn &)
- **gp_ht_map** (const Hash_Fn &, const Eq_Fn &, const Comb_Probe_Fn &, const Probe_Fn &)
- **gp_ht_map** (const Hash_Fn &, const Eq_Fn &, const Comb_Probe_Fn &, const Probe_Fn &, const Resize_Policy &)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- template<typename It>
  void **copy_from_range** (It, It)
- bool empty () const
- iterator **end** ()
- const_iterator **end** () const
- bool **erase** (key_const_reference)
- template<typename Pred>
  size_type **erase_if** (Pred)
- point_iterator **find** (key_const_reference)
- point_const_iterator **find** (key_const_reference) const
- point_iterator **find_end** ()
- point_const_iterator **find_end** () const
- Comb_Probe_Fn & get_comb_probe_fn ()
- const Comb_Probe_Fn & get_comb_probe_fn () const
- Eq_Fn & get_eq_fn ()
- const Eq_Fn & get_eq_fn () const
- Hash_Fn & get_hash_fn ()
- const Hash_Fn & get_hash_fn () const
- Probe_Fn & get_probe_fn ()
- const Probe_Fn & get_probe_fn () const
- Resize_Policy & get_resize_policy ()
- const Resize_Policy & get_resize_policy () const
- std::pair< point_iterator, bool > **insert** (const_reference r_val)
- size_type **max_size** () const
- mapped_reference **operator[ ]** (key_const_reference r_key)
- size_type **size** () const
- void **swap** (gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy > &)

Public Attributes

- no_throw_indicator **m_no_throw_copies_indicator**
- store_extra **m_store_extra_indicator**

Friends

- class **const_iterator_**
- class **iterator_**

5.448.1 Detailed Description

template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
class __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >

A general-probing hash-based container.

Template Parameters

Key	Key type.
Mapped	Map type.
Hash_Fn	Hashing functor. Default is __gnu_cxx::hash.
Eq_Fn	Equal functor. Default std::equal_to<Key>
_Alloc	Allocator type.
Store_Hash	If key type stores extra metadata. Defaults to false.
Comb_Probe_Fn	Combining probe functor. If Hash_Fn is not null_type, then this is the ranged-probe functor; otherwise, this is the range-hashing functor. XXX See Design::Hash-Based Containers::Hash Policies. Default direct_mask_range_hashing.
Probe_Fn	Probe functor. Defaults to linear_probe_fn, also quadratic_probe_fn.
Resize_Policy	Resizes hash. Defaults to hash_standard_resize_policy, using hash_exponential_size_policy and hash_load_check_resize_trigger.

Bases are: detail::hash_eq_fn, Resize_Policy, detail::ranged_probe_fn, detail::types_traits. (Optional: detail::debug_↩ map_base.)

5.448.2 Member Enumeration Documentation

anonymous enum

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
anonymous enum
```

Value stores hash, true or false.

5.448.3 Member Function Documentation

empty()

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
bool __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_↩
Fn, Probe_Fn, Resize_Policy >::empty () const  [inline], [nodiscard]
```

True if size() == 0.

get_comb_probe_fn() [1/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
Comb_Probe_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_comb_probe_fn ()
```

Return current comb_probe_fn.

get_comb_probe_fn() [2/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
const Comb_Probe_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_↩
Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_comb_probe_fn () const
```

Return current const comb_probe_fn.

get_eq_fn() [1/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
Eq_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_↩
Probe_Fn, Probe_Fn, Resize_Policy >::get_eq_fn ()
```

Return current eq_fn.

get_eq_fn() [2/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
const Eq_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_eq_fn () const
```

Return current const eq_fn.

get_hash_fn() [1/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
Hash_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_↩
Probe_Fn, Probe_Fn, Resize_Policy >::get_hash_fn ()
```

Return current hash_fn.

get_hash_fn() [2/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
const Hash_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_hash_fn () const
```

Return current const hash_fn.

get_probe_fn() [1/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
Probe_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_↩
Probe_Fn, Probe_Fn, Resize_Policy >::get_probe_fn ()
```

Return current probe_fn.

get_probe_fn() [2/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
const Probe_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_probe_fn () const
```

Return current const probe_fn.

get_resize_policy() [1/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
Resize_Policy & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_resize_policy ()
```

Return current resize_policy.

get_resize_policy() [2/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
const Resize_Policy & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_↩
Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_resize_policy () const
```

Return current const resize_policy.
The documentation for this class was generated from the following file:

- gp_ht_map_.hpp

5.449 std::chrono::gps_clock Class Reference

`#include <chrono>`

Public Types

- using **duration**
- using **period**
- using **rep**
- using **time_point**

Static Public Member Functions

- template<typename _Duration>
  static gps_time< common_type_t< _Duration, seconds > > **from_utc** (const utc_time< _Duration > &__t)
- static time_point **now** ()
- template<typename _Duration>
  static utc_time< common_type_t< _Duration, seconds > > **to_utc** (const gps_time< _Duration > &__t)

Static Public Attributes

- static constexpr bool **is_steady**

5.449.1 Detailed Description

A clock that measures GPS time.
The epoch is 1980-01-06 00:00:00.

Since

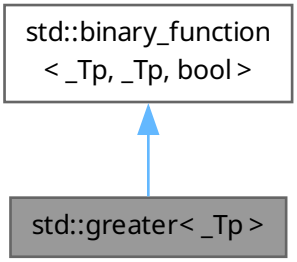
C++20

The documentation for this class was generated from the following file:

- chrono

5.450 std::greater< _Tp > Struct Template Reference

`#include <stl_function.h>`
Inheritance diagram for std::greater< _Tp >:



Public Types

- typedef _Tp first_argument_type
- typedef bool result_type
- typedef _Tp second_argument_type

Public Member Functions

- constexpr bool **operator()** (const _Tp &__x, const _Tp &__y) const

5.450.1 Detailed Description

template<typename _Tp>
struct std::greater< _Tp >

One of the comparison functors.

5.450.2 Member Typedef Documentation

first_argument_type

`typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type [inherited]`
`first_argument_type` is the type of the first argument

result_type

`typedef bool std::binary_function< _Tp, _Tp, bool >::result_type [inherited]`
`result_type` is the return type

second_argument_type

`typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type [inherited]`
`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

5.451 std::ranges::greater Struct Reference

`#include <ranges_cmp.h>`

Public Types

- using **is_transparent**

Public Member Functions

- template<typename _Tp, typename _Up>
  requires totally_ordered_with<_Tp, _Up>
  constexpr bool **operator()** (_Tp &&__t, _Up &&__u) const noexcept(noexcept(std::declval< _Up >()< std::declval< _Tp >()))

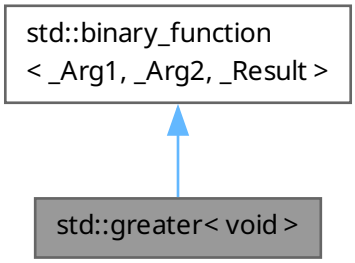
5.451.1 Detailed Description

ranges::greater function object type.
The documentation for this struct was generated from the following file:

- ranges_cmp.h

5.452 std::greater< void > Struct Reference

`#include <stl_function.h>`
Inheritance diagram for std::greater< void >:



Public Types

- typedef void first_argument_type
- typedef __is_transparent **is_transparent**
- typedef bool result_type
- typedef void second_argument_type

Public Member Functions

- template<typename _Tp, typename _Up>
  constexpr auto **operator()** (_Tp &&__t, _Up &&__u) const noexcept(noexcept(std::forward< _Tp >(__t) > std::forward< _Up >(__u))) -> decltype(std::forward< _Tp >(__t) > std::forward< _Up >(__u))

- template<typename _Tp, typename _Up>
  constexpr bool **operator()** (_Tp *__t, _Up *__u) const noexcept
- constexpr bool **operator()** (const void &__x, const void &__y) const

5.452.1 Detailed Description

One of the comparison functors.

5.452.2 Member Typedef Documentation

first_argument_type

```
typedef void std::binary_function< void, void, bool >::first_argument_type
```

`first_argument_type` is the type of the first argument

result_type

```
typedef bool std::binary_function< void, void, bool >::result_type
```

`result_type` is the return type

second_argument_type

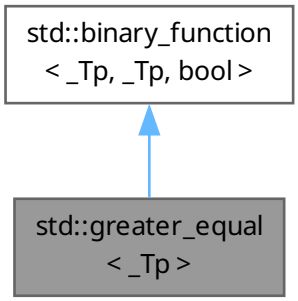
```
typedef void std::binary_function< void, void, bool >::second_argument_type
```

`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

5.453 std::greater_equal< _Tp > Struct Template Reference

`#include <stl_function.h>`
Inheritance diagram for std::greater_equal< _Tp >:



Public Types

- typedef _Tp first_argument_type
- typedef bool result_type
- typedef _Tp second_argument_type

Public Member Functions

- constexpr bool **operator()** (const _Tp &__x, const _Tp &__y) const

5.453.1 Detailed Description

template<**typename _Tp**>
struct std::greater_equal< **_Tp** >

One of the comparison functors.

5.453.2 Member Typedef Documentation

first_argument_type

`typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type [inherited]`
`first_argument_type` is the type of the first argument

result_type

`typedef bool std::binary_function< _Tp, _Tp, bool >::result_type [inherited]`
`result_type` is the return type

second_argument_type

`typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type [inherited]`
`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

5.454 std::ranges::greater_equal Struct Reference

`#include <ranges_cmp.h>`

Public Types

- using **is_transparent**

Public Member Functions

- template<typename _Tp, typename _Up>
  requires totally_ordered_with<_Tp, _Up>
  constexpr bool **operator()** (_Tp &&__t, _Up &&__u) const noexcept(noexcept(std::declval< _Tp >()< std::declval< _Up >()))

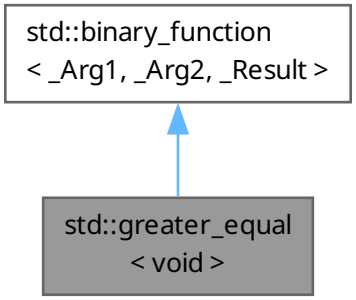
5.454.1 Detailed Description

ranges::greater_equal function object type.
The documentation for this struct was generated from the following file:

- ranges_cmp.h

5.455 std::greater_equal< void > Struct Reference

`#include <stl_function.h>`
Inheritance diagram for std::greater_equal< void >:



Public Types

- typedef void first_argument_type
- typedef __is_transparent **is_transparent**
- typedef bool result_type
- typedef void second_argument_type

Public Member Functions

- template<typename _Tp, typename _Up>
  constexpr auto **operator()** (_Tp &&__t, _Up &&__u) const noexcept(noexcept(std::forward< _Tp >(__t) >=std::forward< _Up >(__u))) -> decltype(std::forward< _Tp >(__t) >=std::forward< _Up >(__u))
- template<typename _Tp, typename _Up>
  constexpr bool **operator()** (_Tp *__t, _Up *__u) const noexcept
- constexpr bool **operator()** (const void &__x, const void &__y) const

5.455.1 Detailed Description

One of the comparison functors.

5.455.2 Member Typedef Documentation

first_argument_type

`typedef void std::binary_function< void, void, bool >::first_argument_type`
`first_argument_type` is the type of the first argument

result_type

`typedef bool std::binary_function< void, void, bool >::result_type`
`result_type` is the return type

second_argument_type

`typedef void std::binary_function< void, void, bool >::second_argument_type`
`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

5.456 __gnu_cxx::random_condition::group_adjustor Struct Reference

`#include <throw_allocator.h>`

Public Member Functions

- **group_adjustor** (size_t size)

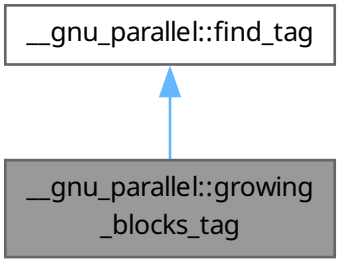
5.456.1 Detailed Description

Group condition.
The documentation for this struct was generated from the following file:

- throw_allocator.h

5.457 __gnu_parallel::growing_blocks_tag Struct Reference

`#include <tags.h>`
Inheritance diagram for __gnu_parallel::growing_blocks_tag:



5.457.1 Detailed Description

Selects the growing block size variant for std::find().

See also

_GLIBCXX_FIND_GROWING_BLOCKS

The documentation for this struct was generated from the following file:

- tags.h

5.458 std::gslice Class Reference

`#include <gslice.h>`

Public Member Functions

- gslice ()
- gslice (const gslice &)
- gslice (size_t __o, const valarray< size_t > &__l, const valarray< size_t > &__s)
- ~gslice ()
- gslice & operator= (const gslice &)
- valarray< size_t > size () const
- size_t start () const
- valarray< size_t > stride () const

Friends

- template<typename _Tp>
  class **valarray**

5.458.1 Detailed Description

Class defining multi-dimensional subset of an array.
The slice class represents a multi-dimensional subset of an array, specified by three parameter sets: start offset, size array, and stride array. The start offset is the index of the first element of the array that is part of the subset. The size and stride array describe each dimension of the slice. Size is the number of elements in that dimension, and stride is the distance in the array between successive elements in that dimension. Each dimension's size and stride is taken to begin at an array element described by the previous dimension. The size array and stride array must be the same size. For example, if you have offset==3, stride[0]==11, size[1]==3, stride[1]==3, then slice[0,0]==array[3], slice[0,1]==array[6], slice[0,2]==array[9], slice[1,0]==array[14], slice[1,1]==array[17], slice[1,2]==array[20].
The documentation for this class was generated from the following file:

- gslice.h

5.459 std::gslice_array< _Tp > Class Template Reference

`#include <gslice_array.h>`

Public Types

- typedef _Tp **value_type**

Public Member Functions

- gslice_array (const gslice_array &)
- template<class _Dom>
  void **operator%=** (const _Expr< _Dom, _Tp > &) const
- void operator%= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator&=** (const _Expr< _Dom, _Tp > &) const
- void operator&= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator*=** (const _Expr< _Dom, _Tp > &) const
- void operator*= (const valarray< _Tp > &) const

- template<class _Dom>
  void **operator+=** (const _Expr< _Dom, _Tp > &) const
- void operator+= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator-=** (const _Expr< _Dom, _Tp > &) const
- void operator-= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator/=** (const _Expr< _Dom, _Tp > &) const
- void operator/= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator<<=** (const _Expr< _Dom, _Tp > &) const
- void operator<<= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator=** (const _Expr< _Dom, _Tp > &) const
- void operator= (const _Tp &) const
- gslice_array & operator= (const gslice_array &)
- void operator= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator>>=** (const _Expr< _Dom, _Tp > &) const
- void operator>>= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator^=** (const _Expr< _Dom, _Tp > &) const
- void operator^= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator|=** (const _Expr< _Dom, _Tp > &) const
- void operator|= (const valarray< _Tp > &) const

Friends

- class **valarray< _Tp >**

5.459.1 Detailed Description

template<typename _Tp>
class std::gslice_array< _Tp >

Reference to multi-dimensional subset of an array.
A gslice_array is a reference to the actual elements of an array specified by a gslice. The way to get a gslice_array is to call operator[](gslice) on a valarray. The returned gslice_array then permits carrying operations out on the referenced subset of elements in the original valarray. For example, operator+=(valarray) will add values to the subset of elements in the underlying valarray this gslice_array refers to.

Parameters

Tp	Element type.

5.459.2 Member Function Documentation

operator%=()

```
template<typename _Tp>
void std::gslice_array< _Tp >::operator%= (
            const valarray< _Tp > & ) const
```

Modulo slice elements by corresponding elements of *v*.

operator&=()

```
template<typename _Tp>
void std::gslice_array< _Tp >::operator&= (
            const valarray< _Tp > & ) const
```

Logical and slice elements with corresponding elements of *v*.

operator*=()

```
template<typename _Tp>
void std::gslice_array< _Tp >::operator*= (
            const valarray< _Tp > & ) const
```

Multiply slice elements by corresponding elements of *v*.

operator+=()

```
template<typename _Tp>
void std::gslice_array< _Tp >::operator+= (
            const valarray< _Tp > & ) const
```

Add corresponding elements of *v* to slice elements.

operator-=()

```
template<typename _Tp>
void std::gslice_array< _Tp >::operator-= (
            const valarray< _Tp > & ) const
```

Subtract corresponding elements of *v* from slice elements.

operator/=()

```
template<typename _Tp>
void std::gslice_array< _Tp >::operator/= (
            const valarray< _Tp > & ) const
```

Divide slice elements by corresponding elements of *v*.

operator<<=()

```
template<typename _Tp>
void std::gslice_array< _Tp >::operator<<= (
            const valarray< _Tp > & ) const
```

Left shift slice elements by corresponding elements of *v*.

operator>>=()

```
template<typename _Tp>
void std::gslice_array< _Tp >::operator>>= (
            const valarray< _Tp > & ) const
```

Right shift slice elements by corresponding elements of *v*.
References gslice_array().

operator^=()

```
template<typename _Tp>
void std::gslice_array< _Tp >::operator^= (
            const valarray< _Tp > & ) const
```

Logical xor slice elements with corresponding elements of *v*.

operator"" |=()

```
template<typename _Tp>
void std::gslice_array< _Tp >::operator|= (
            const valarray< _Tp > & ) const
```

Logical or slice elements with corresponding elements of *v*.
The documentation for this class was generated from the following files:

- valarray
- gslice_array.h

5.460 std::has_virtual_destructor< _Tp > Struct Template Reference

```
#include <type_traits>
```

5.460.1 Detailed Description

template<typename _Tp>
struct std::has_virtual_destructor< _Tp >

has_virtual_destructor
The documentation for this struct was generated from the following file:

- type_traits

5.461 std::hash< _Tp > Struct Template Reference

```
#include <functional_hash.h>
```

5.461.1 Detailed Description

template<typename _Tp>
struct std::hash< _Tp >

Primary class template hash.
Primary class template hash, usable for enum types only.
The documentation for this struct was generated from the following file:

- string_view

5.462 std::hash< __debug::bitset< _Nb > > Struct Template Reference

```
#include <bitset>
```

Public Member Functions

- size_t **operator()** (const __debug::bitset< _Nb > &__b) const noexcept

5.462.1 Detailed Description

template<size_t _Nb>
struct std::hash< __debug::bitset< _Nb > >

std::hash specialization for bitset.
The documentation for this struct was generated from the following file:

- debug/bitset

5.463 std::hash< __debug::vector< bool, _Alloc > > Struct Template Reference

`#include <vector>`

Public Member Functions

- size_t **operator()** (const __debug::vector< bool, _Alloc > &__b) const noexcept

5.463.1 Detailed Description

template<typename _Alloc>
struct std::hash< __debug::vector< bool, _Alloc > >

std::hash specialization for vector<bool>.
The documentation for this struct was generated from the following file:

- debug/vector

5.464 std::hash< __gnu_cxx::__u16vstring > Struct Reference

`#include <vstring.h>`

Public Member Functions

- size_t **operator()** (const __gnu_cxx::__u16vstring &__s) const noexcept

5.464.1 Detailed Description

std::hash specialization for __u16vstring.
The documentation for this struct was generated from the following file:

- vstring.h

5.465 std::hash< __gnu_cxx::__u32vstring > Struct Reference

`#include <vstring.h>`

Public Member Functions

- size_t **operator()** (const __gnu_cxx::__u32vstring &__s) const noexcept

5.465.1 Detailed Description

std::hash specialization for __u32vstring.
The documentation for this struct was generated from the following file:

- vstring.h

5.466 std::hash< __gnu_cxx::__vstring > Struct Reference

`#include <vstring.h>`

Public Member Functions

- size_t **operator()** (const __gnu_cxx::__vstring &__s) const noexcept

5.466.1 Detailed Description

std::hash specialization for __vstring.
The documentation for this struct was generated from the following file:

- vstring.h

5.467 std::hash< __gnu_cxx::__wvstring > Struct Reference

`#include <vstring.h>`

Public Member Functions

- size_t **operator()** (const __gnu_cxx::__wvstring &__s) const noexcept

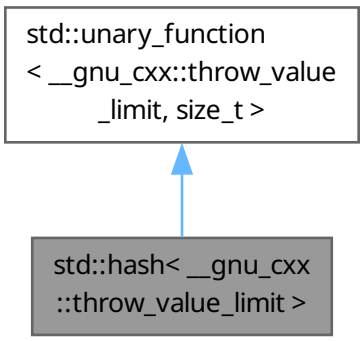
5.467.1 Detailed Description

std::hash specialization for __wvstring.
The documentation for this struct was generated from the following file:

- vstring.h

5.468 std::hash< __gnu_cxx::throw_value_limit > Struct Reference

`#include <throw_allocator.h>`
Inheritance diagram for std::hash< __gnu_cxx::throw_value_limit >:



Public Types

- typedef __gnu_cxx::throw_value_limit argument_type
- typedef size_t result_type

Public Member Functions

- size_t **operator()** (const __gnu_cxx::throw_value_limit &__val) const

5.468.1 Detailed Description

Explicit specialization of std::hash for __gnu_cxx::throw_value_limit.

5.468.2 Member Typedef Documentation

argument_type

```
typedef __gnu_cxx::throw_value_limit std::unary_function< __gnu_cxx::throw_value_limit, size_↩
t >::argument_type [inherited]
```

`argument_type` is the type of the argument

result_type

```
typedef size_t std::unary_function< __gnu_cxx::throw_value_limit, size_t >::result_type [inherited]
```

`result_type` is the return type

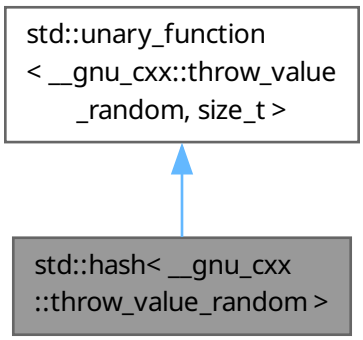
The documentation for this struct was generated from the following file:

- throw_allocator.h

5.469 std::hash< __gnu_cxx::throw_value_random > Struct Reference

`#include <throw_allocator.h>`

Inheritance diagram for std::hash< __gnu_cxx::throw_value_random >:



Public Types

- typedef __gnu_cxx::throw_value_random argument_type
- typedef size_t result_type

Public Member Functions

- size_t **operator()** (const __gnu_cxx::throw_value_random &__val) const

5.469.1 Detailed Description

Explicit specialization of std::hash for __gnu_cxx::throw_value_random.

5.469.2 Member Typedef Documentation

argument_type

```
typedef __gnu_cxx::throw_value_random std::unary_function< __gnu_cxx::throw_value_random, size_t
>::argument_type [inherited]
```

`argument_type` is the type of the argument

result_type

```
typedef size_t std::unary_function< __gnu_cxx::throw_value_random, size_t >::result_type [inherited]
```

`result_type` is the return type

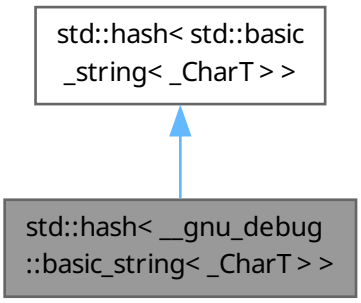
The documentation for this struct was generated from the following file:

- throw_allocator.h

5.470 std::hash< __gnu_debug::basic_string< _CharT > > Struct Template Reference

`#include <string>`

Inheritance diagram for std::hash< __gnu_debug::basic_string< _CharT > >:



5.470.1 Detailed Description

template<typename _CharT>
struct std::hash< __gnu_debug::basic_string< _CharT > >

std::hash specialization for __gnu_debug::basic_string.

The documentation for this struct was generated from the following file:

- debug/string

5.471 std::hash< __shared_ptr< _Tp, _Lp > > Struct Template Reference

`#include <shared_ptr_base.h>`

Public Member Functions

- size_t **operator()** (const __shared_ptr< _Tp, _Lp > &__s) const noexcept

5.471.1 Detailed Description

template<typename _Tp, _Lock_policy _Lp>
struct std::hash< __shared_ptr< _Tp, _Lp > >

std::hash specialization for __shared_ptr.
The documentation for this struct was generated from the following file:

- shared_ptr_base.h

5.472 std::hash< _Tp * > Struct Template Reference

`#include <functional_hash.h>`

Public Member Functions

- size_t **operator()** (_Tp *__p) const noexcept

5.472.1 Detailed Description

template<typename _Tp>
struct std::hash< _Tp * >

Partial specializations for pointer types.
The documentation for this struct was generated from the following file:

- functional_hash.h

5.473 std::hash< basic_string< char, char_traits< char >, _Alloc > > Struct Template Reference

`#include <basic_string.h>`

Public Member Functions

- size_t **operator()** (const _StrT &__s) const noexcept

5.473.1 Detailed Description

template<typename _Alloc>
struct std::hash< basic_string< char, char_traits< char >, _Alloc > >

std::hash specialization for string.
The documentation for this struct was generated from the following file:

- basic_string.h

5.474 std::hash< basic_string< char16_t, char_traits< char16_t >, _Alloc > > Struct Template Reference

`#include <basic_string.h>`

Public Member Functions

- size_t **operator()** (const _StrT &__s) const noexcept

5.474.1 Detailed Description

template<**typename _Alloc**>
struct std::hash< **basic_string**< **char16_t, char_traits**< **char16_t** >, **_Alloc** > >

std::hash specialization for u16string.
The documentation for this struct was generated from the following file:

- basic_string.h

5.475 std::hash< basic_string< char32_t, char_traits< char32_t >, _Alloc > > Struct Template Reference

`#include <basic_string.h>`

Public Member Functions

- size_t **operator()** (const _StrT &__s) const noexcept

5.475.1 Detailed Description

template<**typename _Alloc**>
struct std::hash< **basic_string**< **char32_t, char_traits**< **char32_t** >, **_Alloc** > >

std::hash specialization for u32string.
The documentation for this struct was generated from the following file:

- basic_string.h

5.476 std::hash< basic_string< wchar_t, char_traits< wchar_t >, _Alloc > > Struct Template Reference

`#include <basic_string.h>`

Public Member Functions

- size_t **operator()** (const _StrT &__s) const noexcept

5.476.1 Detailed Description

template<**typename _Alloc**>
struct std::hash< **basic_string**< **wchar_t, char_traits**< **wchar_t** >, **_Alloc** > >

std::hash specialization for wstring.
The documentation for this struct was generated from the following file:

- basic_string.h

5.477 std::hash< bool > Struct Reference

`#include <functional_hash.h>`

Public Member Functions

- size_t **operator()** (bool __val) const noexcept

5.477.1 Detailed Description

Explicit specialization for bool.
The documentation for this struct was generated from the following file:

- functional_hash.h

5.478 std::hash< char > Struct Reference

`#include <functional_hash.h>`

Public Member Functions

- size_t **operator()** (char __val) const noexcept

5.478.1 Detailed Description

Explicit specialization for char.
The documentation for this struct was generated from the following file:

- functional_hash.h

5.479 std::hash< char16_t > Struct Reference

`#include <functional_hash.h>`

Public Member Functions

- size_t **operator()** (char16_t __val) const noexcept

5.479.1 Detailed Description

Explicit specialization for char16_t.
The documentation for this struct was generated from the following file:

- functional_hash.h

5.480 std::hash< char32_t > Struct Reference

`#include <functional_hash.h>`

Public Member Functions

- size_t **operator()** (char32_t __val) const noexcept

5.480.1 Detailed Description

Explicit specialization for char32_t.
The documentation for this struct was generated from the following file:

- functional_hash.h

5.481 std::hash< double > Struct Reference

`#include <functional_hash.h>`

Public Member Functions

- size_t **operator()** (double __val) const noexcept

5.481.1 Detailed Description

Specialization for double.
The documentation for this struct was generated from the following file:

- functional_hash.h

5.482 std::hash< error_code > Struct Reference

`#include <system_error>`

Public Member Functions

- size_t **operator()** (const error_code &__e) const noexcept

5.482.1 Detailed Description

std::hash specialization for error_code.
The documentation for this struct was generated from the following file:

- system_error

5.483 std::hash< error_condition > Struct Reference

`#include <system_error>`

Public Member Functions

- size_t **operator()** (const error_condition &__e) const noexcept

5.483.1 Detailed Description

std::hash specialization for error_condition.
The documentation for this struct was generated from the following file:

- system_error

5.484 std::hash< experimental::optional< _Tp > > Struct Template Reference

`#include <optional>`

Public Types

- using **argument_type**
- using **result_type**

Public Member Functions

- size_t **operator()** (const experimental::optional< _Tp > &__t) const noexcept(noexcept(hash< _Tp > {}(*__t)))

5.484.1 Detailed Description

template<typename _Tp>
struct std::hash< experimental::optional< _Tp > >

std::hash partial specialization for experimental::optional
The documentation for this struct was generated from the following file:

- experimental/optional

5.485 std::hash< experimental::shared_ptr< _Tp > > Struct Template Reference

`#include <shared_ptr.h>`

Public Member Functions

- size_t **operator()** (const experimental::shared_ptr< _Tp > &__s) const noexcept

5.485.1 Detailed Description

template<typename _Tp>
struct std::hash< experimental::shared_ptr< _Tp > >

std::hash specialization for shared_ptr.
The documentation for this struct was generated from the following file:

- experimental/bits/shared_ptr.h

5.486 std::hash< float > Struct Reference

`#include <functional_hash.h>`

Public Member Functions

- size_t **operator()** (float __val) const noexcept

5.486.1 Detailed Description

Specialization for float.
The documentation for this struct was generated from the following file:

- functional_hash.h

5.487 std::hash< int > Struct Reference

`#include <functional_hash.h>`

Public Member Functions

- size_t **operator()** (int __val) const noexcept

5.487.1 Detailed Description

Explicit specialization for int.
The documentation for this struct was generated from the following file:

- functional_hash.h

5.488 std::hash< long > Struct Reference

`#include <functional_hash.h>`

Public Member Functions

- size_t **operator()** (long __val) const noexcept

5.488.1 Detailed Description

Explicit specialization for long.
The documentation for this struct was generated from the following file:

- functional_hash.h

5.489 std::hash< long double > Struct Reference

`#include <functional_hash.h>`

Public Member Functions

- size_t **operator()** (long double __val) const noexcept

5.489.1 Detailed Description

Specialization for long double.
The documentation for this struct was generated from the following file:

- functional_hash.h

5.490 std::hash< long long > Struct Reference

`#include <functional_hash.h>`

Public Member Functions

- size_t **operator()** (long long __val) const noexcept

5.490.1 Detailed Description

Explicit specialization for long long.
The documentation for this struct was generated from the following file:

- functional_hash.h

5.491 std::hash< shared_ptr< _Tp > > Struct Template Reference

`#include <shared_ptr.h>`

Public Member Functions

- size_t **operator()** (const shared_ptr< _Tp > &__s) const noexcept

5.491.1 Detailed Description

template<typename _Tp>
struct std::hash< shared_ptr< _Tp > >

std::hash specialization for shared_ptr.
The documentation for this struct was generated from the following file:

- bits/shared_ptr.h

5.492 std::hash< short > Struct Reference

`#include <functional_hash.h>`

Public Member Functions

- size_t **operator()** (short __val) const noexcept

5.492.1 Detailed Description

Explicit specialization for short.
The documentation for this struct was generated from the following file:

- functional_hash.h

5.493 std::hash< signed char > Struct Reference

`#include <functional_hash.h>`

Public Member Functions

- size_t **operator()** (signed char __val) const noexcept

5.493.1 Detailed Description

Explicit specialization for signed char.
The documentation for this struct was generated from the following file:

- functional_hash.h

5.494 std::hash< thread::id > Struct Reference

`#include <std_thread.h>`

Public Member Functions

- size_t **operator()** (const thread::id &__id) const noexcept

5.494.1 Detailed Description

std::hash specialization for thread::id.
The documentation for this struct was generated from the following file:

- std_thread.h

5.495 std::hash< type_index > Struct Reference

`#include <typeindex>`

Public Types

- typedef type_index **argument_type**
- typedef size_t **result_type**

Public Member Functions

- size_t **operator()** (const type_index &__ti) const noexcept

5.495.1 Detailed Description

std::hash specialization for type_index.
The documentation for this struct was generated from the following file:

- typeindex

5.496 std::hash< unique_ptr< _Tp, _Dp > > Struct Template Reference

`#include <unique_ptr.h>`

5.496.1 Detailed Description

template<typename _Tp, typename _Dp>
struct std::hash< unique_ptr< _Tp, _Dp > >

std::hash specialization for unique_ptr.
The documentation for this struct was generated from the following file:

- unique_ptr.h

5.497 std::hash< unsigned char > Struct Reference

`#include <functional_hash.h>`

Public Member Functions

- size_t **operator()** (unsigned char __val) const noexcept

5.497.1 Detailed Description

Explicit specialization for unsigned char.
The documentation for this struct was generated from the following file:

- functional_hash.h

5.498 std::hash< unsigned int > Struct Reference

`#include <functional_hash.h>`

Public Member Functions

- size_t **operator()** (unsigned int __val) const noexcept

5.498.1 Detailed Description

Explicit specialization for unsigned int.
The documentation for this struct was generated from the following file:

- functional_hash.h

5.499 std::hash< unsigned long > Struct Reference

`#include <functional_hash.h>`

Public Member Functions

- size_t **operator()** (unsigned long __val) const noexcept

5.499.1 Detailed Description

Explicit specialization for unsigned long.
The documentation for this struct was generated from the following file:

- functional_hash.h

5.500 std::hash< unsigned long long > Struct Reference

`#include <functional_hash.h>`

Public Member Functions

- size_t **operator()** (unsigned long long __val) const noexcept

5.500.1 Detailed Description

Explicit specialization for unsigned long long.
The documentation for this struct was generated from the following file:

- functional_hash.h

5.501 std::hash< unsigned short > Struct Reference

`#include <functional_hash.h>`

Public Member Functions

- size_t **operator()** (unsigned short __val) const noexcept

5.501.1 Detailed Description

Explicit specialization for unsigned short.
The documentation for this struct was generated from the following file:

- functional_hash.h

5.502 std::hash< wchar_t > Struct Reference

`#include <functional_hash.h>`

Public Member Functions

- size_t **operator()** (wchar_t __val) const noexcept

5.502.1 Detailed Description

Explicit specialization for wchar_t.
The documentation for this struct was generated from the following file:

- functional_hash.h

5.503 std::hash<::bitset< _Nb > > Struct Template Reference

`#include <bitset>`

Public Member Functions

- size_t **operator()** (const ::bitset< _Nb > &__b) const noexcept

5.503.1 Detailed Description

template<size_t _Nb>
struct std::hash<::bitset< _Nb > >

std::hash specialization for bitset.
The documentation for this struct was generated from the following file:

- bitset

5.504 std::hash<::vector< bool, _Alloc > > Struct Template Reference

`#include <stl_bvector.h>`

Public Member Functions

- size_t **operator()** (const ::vector< bool, _Alloc > &) const noexcept

5.504.1 Detailed Description

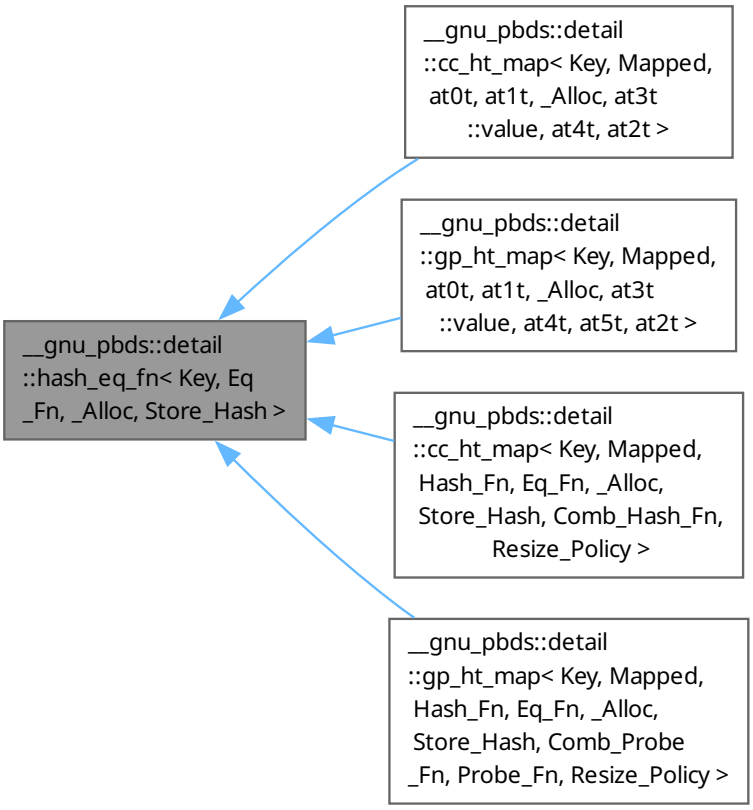
template<typename _Alloc>
struct std::hash<::vector< bool, _Alloc > >

std::hash specialization for vector<bool>.
The documentation for this struct was generated from the following files:

- stl_bvector.h
- vector.tcc

5.505 __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, Store_Hash > Struct Template Reference

Inheritance diagram for __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, Store_Hash >:



5.505.1 Detailed Description

template<typename Key, typename Eq_Fn, typename _Alloc, bool Store_Hash>
struct __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, Store_Hash >

Primary template.
The documentation for this struct was generated from the following file:

- hash_eq_fn.hpp

5.506 __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, false > Struct Template Reference

```
#include <hash_eq_fn.hpp>
```

Public Types

- typedef Eq_Fn **eq_fn_base**

- typedef rebind_traits< _Alloc, Key >::const_reference **key_const_reference**

Public Member Functions

- **hash_eq_fn** (const Eq_Fn &r_eq_fn)
- bool **operator()** (key_const_reference r_lhs_key, key_const_reference r_rhs_key) const
- void **swap** (const hash_eq_fn &other)

5.506.1 Detailed Description

template<typename Key, typename Eq_Fn, typename _Alloc>
struct __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, false >

Specialization 1 - The client requests that hash values not be stored.
The documentation for this struct was generated from the following file:

- hash_eq_fn.hpp

5.507 __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, true > Struct Template Reference

`#include <hash_eq_fn.hpp>`

Public Types

- typedef Eq_Fn **eq_fn_base**
- typedef rebind_traits< _Alloc, Key >::const_reference **key_const_reference**
- typedef _Alloc::size_type **size_type**

Public Member Functions

- **hash_eq_fn** (const Eq_Fn &r_eq_fn)
- bool **operator()** (key_const_reference r_lhs_key, size_type lhs_hash, key_const_reference r_rhs_key, size_type rhs_hash) const
- void **swap** (const hash_eq_fn &other)

5.507.1 Detailed Description

template<typename Key, class Eq_Fn, class _Alloc>
struct __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, true >

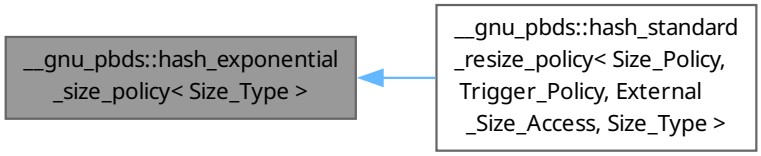
Specialization 2 - The client requests that hash values be stored.
The documentation for this struct was generated from the following file:

- hash_eq_fn.hpp

5.508 __gnu_pbds::hash_exponential_size_policy< Size_Type > Class Template Reference

`#include <hash_policy.hpp>`

Inheritance diagram for __gnu_pbds::hash_exponential_size_policy< Size_Type >:



Public Types

- typedef Size_Type **size_type**

Public Member Functions

- hash_exponential_size_policy (size_type start_size=8, size_type grow_factor=2)
- void **swap** (hash_exponential_size_policy< Size_Type > &other)

Protected Member Functions

- size_type **get_nearest_larger_size** (size_type size) const
- size_type **get_nearest_smaller_size** (size_type size) const

5.508.1 Detailed Description

template<typename Size_Type = std::size_t>
class __gnu_pbds::hash_exponential_size_policy< Size_Type >

A size policy whose sequence of sizes form an exponential sequence (typically powers of 2.

5.508.2 Constructor & Destructor Documentation

hash_exponential_size_policy()

```
template<typename Size_Type>
__gnu_pbds::hash_exponential_size_policy< Size_Type >::hash_exponential_size_policy (
            size_type start_size = 8,
            size_type grow_factor = 2)
```

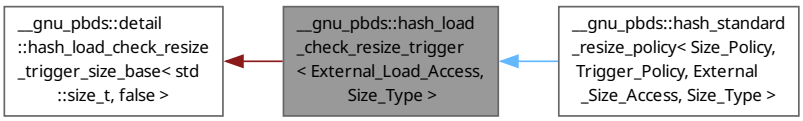
Default constructor, or onstructor taking a start_size, or constructor taking a start size and grow_factor. The policy will use the sequence of sizes start_size, start_size∗ grow_factor, start_size∗ grow_factor^2, ...
The documentation for this class was generated from the following file:

- hash_policy.hpp

5.509 __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type > Class Template Reference

```
#include <hash_policy.hpp>
```

Inheritance diagram for __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >:



Public Types

- enum { external_load_access }
- typedef Size_Type **size_type**

Public Member Functions

- hash_load_check_resize_trigger (float load_min=0.125, float load_max=0.5)
- std::pair< float, float > get_loads () const
- void set_loads (std::pair< float, float > load_pair)
- void **swap** (hash_load_check_resize_trigger &other)

Protected Member Functions

- bool **is_grow_needed** (size_type size, size_type num_entries) const
- bool **is_resize_needed** () const
- void notify_cleared ()
- void **notify_erase_search_collision** ()
- void **notify_erase_search_end** ()
- void **notify_erase_search_start** ()
- void **notify_erased** (size_type num_entries)
- void **notify_externally_resized** (size_type new_size)
- void **notify_find_search_collision** ()
- void **notify_find_search_end** ()
- void **notify_find_search_start** ()
- void **notify_insert_search_collision** ()
- void **notify_insert_search_end** ()
- void **notify_insert_search_start** ()
- void notify_inserted (size_type num_entries)
- void notify_resized (size_type new_size)

5.509.1 Detailed Description

template<bool External_Load_Access = false, typename Size_Type = std::size_t>
class __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >

A resize trigger policy based on a load check. It keeps the load factor between some load factors load_min and load↩
_max.

5.509.2 Member Enumeration Documentation

anonymous enum

```
template<bool External_Load_Access = false, typename Size_Type = std::size_t>
anonymous enum
```

Enumerator

external_load_access	Specifies whether the load factor can be accessed externally. The two options have different trade-offs in terms of flexibility, genericity, and encapsulation.

5.509.3 Constructor & Destructor Documentation

hash_load_check_resize_trigger()

```
template<bool External_Load_Access, typename Size_Type>
__gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::hash_load_check_↩
resize_trigger (
            float load_min = 0.125,
            float load_max = 0.5)
```

Default constructor, or constructor taking load_min and load_max load factors between which this policy will keep the actual load.

5.509.4 Member Function Documentation

get_loads()

```
template<bool External_Load_Access, typename Size_Type>
std::pair< float, float > __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size↩
_Type >::get_loads () const  [inline]
```

Returns a pair of the minimal and maximal loads, respectively.

notify_cleared()

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::notify_↩
cleared ()  [protected]
```

Notifies the table was cleared.

notify_inserted()

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::notify_↩
inserted (
            size_type num_entries)  [inline], [protected]
```

Notifies an element was inserted. The total number of entries in the table is num_entries.

notify_resized()

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::notify_↩
resized (
            size_type new_size)  [protected]
```

Notifies the table was resized as a result of this object's signifying that a resize is needed.

References notify_resized().
Referenced by notify_resized().

set_loads()

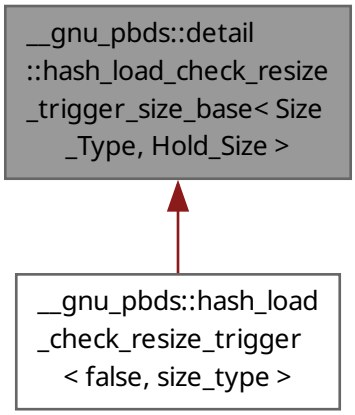
```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::set_loads (
            std::pair< float, float > load_pair)
```

Sets the loads through a pair of the minimal and maximal loads, respectively.
The documentation for this class was generated from the following file:

- hash_policy.hpp

5.510 __gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, Hold_Size > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, Hold_Size >:



5.510.1 Detailed Description

template<typename Size_Type, bool Hold_Size>
class __gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, Hold_Size >

Primary template.
The documentation for this class was generated from the following file:

- hash_load_check_resize_trigger_size_base.hpp

5.511 __gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, true > Class Template Reference

```
#include <hash_load_check_resize_trigger_size_base.hpp>
```

Protected Types

- typedef Size_Type **size_type**

Protected Member Functions

- size_type **get_size** () const
- void **set_size** (size_type size)
- void **swap** (hash_load_check_resize_trigger_size_base &other)

5.511.1 Detailed Description

template<typename Size_Type>
class __gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, true >

Specializations.
The documentation for this class was generated from the following file:

- hash_load_check_resize_trigger_size_base.hpp

5.512 __gnu_cxx::hash_map< _Key, _Tp, _HashFn, _EqualKey, _Alloc > Class Template Reference

`#include <hash_map>`

Public Types

- typedef _Ht::allocator_type **allocator_type**
- typedef _Ht::const_iterator **const_iterator**
- typedef _Ht::const_pointer **const_pointer**
- typedef _Ht::const_reference **const_reference**
- typedef _Tp **data_type**
- typedef _Ht::difference_type **difference_type**
- typedef _Ht::hasher **hasher**
- typedef _Ht::iterator **iterator**
- typedef _Ht::key_equal **key_equal**
- typedef _Ht::key_type **key_type**
- typedef _Tp **mapped_type**
- typedef _Ht::pointer **pointer**
- typedef _Ht::reference **reference**
- typedef _Ht::size_type **size_type**
- typedef _Ht::value_type **value_type**

Public Member Functions

- template<class _InputIterator>
  hash_map (_InputIterator __f, _InputIterator __l)
- template<class _InputIterator>
  hash_map (_InputIterator __f, _InputIterator __l, size_type __n)
- template<class _InputIterator>
  hash_map (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf)
- template<class _InputIterator>
  hash_map (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf, const key_equal &__eql, const allocator_type &__a=allocator_type())

- **hash_map** (size_type __n)
- **hash_map** (size_type __n, const hasher &__hf)
- **hash_map** (size_type __n, const hasher &__hf, const key_equal &__eql, const allocator_type &__a=allocator_type())
- iterator **begin** ()
- const_iterator **begin** () const
- size_type **bucket_count** () const
- void **clear** ()
- size_type **count** (const key_type &__key) const
- size_type **elems_in_bucket** (size_type __n) const
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- pair< iterator, iterator > **equal_range** (const key_type &__key)
- pair< const_iterator, const_iterator > **equal_range** (const key_type &__key) const
- size_type **erase** (const key_type &__key)
- void **erase** (iterator __f, iterator __l)
- void **erase** (iterator __it)
- iterator **find** (const key_type &__key)
- const_iterator **find** (const key_type &__key) const
- allocator_type **get_allocator** () const
- hasher **hash_funct** () const
- template<class _InputIterator>
  void **insert** (_InputIterator __f, _InputIterator __l)
- pair< iterator, bool > **insert** (const value_type &__obj)
- pair< iterator, bool > **insert_noresize** (const value_type &__obj)
- key_equal **key_eq** () const
- size_type **max_bucket_count** () const
- size_type **max_size** () const
- _Tp & **operator[ ]** (const key_type &__key)
- void **resize** (size_type __hint)
- size_type **size** () const
- void **swap** (hash_map &__hs)

Friends

- template<class _K1, class _T1, class _HF, class _EqK, class _Al>
  bool **operator==** (const hash_map< _K1, _T1, _HF, _EqK, _Al > &, const hash_map< _K1, _T1, _HF, _EqK, _Al > &)

5.512.1 Detailed Description

template<class _Key, class _Tp, class _HashFn = hash<_Key>, class _EqualKey = equal_to<_Key>, class _Alloc = allocator<_Tp>>
class __gnu_cxx::hash_map< _Key, _Tp, _HashFn, _EqualKey, _Alloc >

This is an SGI extension.

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

The documentation for this class was generated from the following file:

- hash_map

5.513 __gnu_cxx::hash_multimap< _Key, _Tp, _HashFn, _EqualKey, _Alloc > Class Template Reference

`#include <hash_map>`

Public Types

- typedef _Ht::allocator_type **allocator_type**
- typedef _Ht::const_iterator **const_iterator**
- typedef _Ht::const_pointer **const_pointer**
- typedef _Ht::const_reference **const_reference**
- typedef _Tp **data_type**
- typedef _Ht::difference_type **difference_type**
- typedef _Ht::hasher **hasher**
- typedef _Ht::iterator **iterator**
- typedef _Ht::key_equal **key_equal**
- typedef _Ht::key_type **key_type**
- typedef _Tp **mapped_type**
- typedef _Ht::pointer **pointer**
- typedef _Ht::reference **reference**
- typedef _Ht::size_type **size_type**
- typedef _Ht::value_type **value_type**

Public Member Functions

- template<class _InputIterator>
  hash_multimap (_InputIterator __f, _InputIterator __l)
- template<class _InputIterator>
  hash_multimap (_InputIterator __f, _InputIterator __l, size_type __n)
- template<class _InputIterator>
  hash_multimap (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf)
- template<class _InputIterator>
  hash_multimap (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf, const key_equal &_↩
  _eql, const allocator_type &__a=allocator_type())
- **hash_multimap** (size_type __n)
- **hash_multimap** (size_type __n, const hasher &__hf)
- **hash_multimap** (size_type __n, const hasher &__hf, const key_equal &__eql, const allocator_type &__↩
  a=allocator_type())
- iterator **begin** ()
- const_iterator **begin** () const
- size_type **bucket_count** () const
- void **clear** ()
- size_type **count** (const key_type &__key) const
- size_type **elems_in_bucket** (size_type __n) const
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- pair< iterator, iterator > **equal_range** (const key_type &__key)
- pair< const_iterator, const_iterator > **equal_range** (const key_type &__key) const
- size_type **erase** (const key_type &__key)
- void **erase** (iterator __f, iterator __l)
- void **erase** (iterator __it)

- iterator **find** (const key_type &__key)
- const_iterator **find** (const key_type &__key) const
- allocator_type **get_allocator** () const
- hasher **hash_funct** () const
- template<class _InputIterator>
  void **insert** (_InputIterator __f, _InputIterator __l)
- iterator **insert** (const value_type &__obj)
- iterator **insert_noresize** (const value_type &__obj)
- key_equal **key_eq** () const
- size_type **max_bucket_count** () const
- size_type **max_size** () const
- void **resize** (size_type __hint)
- size_type **size** () const
- void **swap** (hash_multimap &__hs)

Friends

- template<class _K1, class _T1, class _HF, class _EqK, class _Al>
  bool **operator==** (const hash_multimap< _K1, _T1, _HF, _EqK, _Al > &, const hash_multimap< _K1, _T1, _HF, _EqK, _Al > &)

5.513.1 Detailed Description

template<class _Key, class _Tp, class _HashFn = hash<_Key>, class _EqualKey = equal_to<_Key>, class _Alloc = allocator<_Tp>>
class __gnu_cxx::hash_multimap< _Key, _Tp, _HashFn, _EqualKey, _Alloc >

This is an SGI extension.

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

The documentation for this class was generated from the following file:

- hash_map

5.514 __gnu_cxx::hash_multiset< _Value, _HashFcn, _EqualKey, _Alloc > Class Template Reference

`#include <hash_set>`

Public Types

- typedef _Ht::allocator_type **allocator_type**
- typedef _Ht::const_iterator **const_iterator**
- typedef _Alloc::const_pointer **const_pointer**
- typedef _Alloc::const_reference **const_reference**
- typedef _Ht::difference_type **difference_type**
- typedef _Ht::hasher **hasher**
- typedef _Ht::const_iterator **iterator**
- typedef _Ht::key_equal **key_equal**
- typedef _Ht::key_type **key_type**
- typedef _Alloc::pointer **pointer**
- typedef _Alloc::reference **reference**
- typedef _Ht::size_type **size_type**
- typedef _Ht::value_type **value_type**

Public Member Functions

- template<class _InputIterator>
  hash_multiset (_InputIterator __f, _InputIterator __l)
- template<class _InputIterator>
  hash_multiset (_InputIterator __f, _InputIterator __l, size_type __n)
- template<class _InputIterator>
  hash_multiset (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf)
- template<class _InputIterator>
  hash_multiset (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf, const key_equal &__eql, const allocator_type &__a=allocator_type())
- **hash_multiset** (size_type __n)
- **hash_multiset** (size_type __n, const hasher &__hf)
- **hash_multiset** (size_type __n, const hasher &__hf, const key_equal &__eql, const allocator_type &__a=allocator_type())
- iterator **begin** () const
- size_type **bucket_count** () const
- void **clear** ()
- size_type **count** (const key_type &__key) const
- size_type **elems_in_bucket** (size_type __n) const
- bool **empty** () const
- iterator **end** () const
- pair< iterator, iterator > **equal_range** (const key_type &__key) const
- size_type **erase** (const key_type &__key)
- void **erase** (iterator __f, iterator __l)
- void **erase** (iterator __it)
- iterator **find** (const key_type &__key) const
- allocator_type **get_allocator** () const
- hasher **hash_funct** () const
- template<class _InputIterator>
  void **insert** (_InputIterator __f, _InputIterator __l)
- iterator **insert** (const value_type &__obj)
- iterator **insert_noresize** (const value_type &__obj)
- key_equal **key_eq** () const
- size_type **max_bucket_count** () const
- size_type **max_size** () const
- void **resize** (size_type __hint)
- size_type **size** () const
- void **swap** (hash_multiset &hs)

Friends

- template<class _Val, class _HF, class _EqK, class _Al>
  bool **operator==** (const hash_multiset< _Val, _HF, _EqK, _Al > &, const hash_multiset< _Val, _HF, _EqK, _Al > &)

5.514.1 Detailed Description

template<class _Value, class _HashFcn = hash<_Value>, class _EqualKey = equal_to<_Value>, class _Alloc = allocator<_Value>>
class __gnu_cxx::hash_multiset< _Value, _HashFcn, _EqualKey, _Alloc >

This is an SGI extension.

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

The documentation for this class was generated from the following file:

- hash_set

5.515 __gnu_pbds::hash_prime_size_policy Class Reference

`#include <hash_policy.hpp>`

Public Types

- typedef std::size_t size_type

Public Member Functions

- hash_prime_size_policy (size_type start_size=8)
- void **swap** (hash_prime_size_policy &other)

Protected Member Functions

- size_type **get_nearest_larger_size** (size_type size) const
- size_type **get_nearest_smaller_size** (size_type size) const

5.515.1 Detailed Description

A size policy whose sequence of sizes form a nearly-exponential sequence of primes.

5.515.2 Member Typedef Documentation

size_type

```
typedef std::size_t __gnu_pbds::hash_prime_size_policy::size_type
```

Size type.

5.515.3 Constructor & Destructor Documentation

hash_prime_size_policy()

```
__gnu_pbds::hash_prime_size_policy::hash_prime_size_policy (
            size_type start_size = 8)  [inline]
```

Default constructor, or onstructor taking a start_size The policy will use the sequence of sizes approximately start_size, start_size∗ 2, start_size∗ 2^2, ...

The documentation for this class was generated from the following file:

- hash_policy.hpp

5.516 __gnu_cxx::hash_set< _Value, _HashFcn, _EqualKey, _Alloc > Class Template Reference

`#include <hash_set>`

Public Types

- typedef _Ht::allocator_type **allocator_type**
- typedef _Ht::const_iterator **const_iterator**
- typedef _Alloc_traits::const_pointer **const_pointer**
- typedef _Alloc_traits::const_reference **const_reference**
- typedef _Ht::difference_type **difference_type**
- typedef _Ht::hasher **hasher**
- typedef _Ht::const_iterator **iterator**
- typedef _Ht::key_equal **key_equal**
- typedef _Ht::key_type **key_type**
- typedef _Alloc_traits::pointer **pointer**
- typedef _Alloc_traits::reference **reference**
- typedef _Ht::size_type **size_type**
- typedef _Ht::value_type **value_type**

Public Member Functions

- template<class _InputIterator>
  hash_set (_InputIterator __f, _InputIterator __l)
- template<class _InputIterator>
  hash_set (_InputIterator __f, _InputIterator __l, size_type __n)
- template<class _InputIterator>
  hash_set (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf)
- template<class _InputIterator>
  hash_set (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf, const key_equal &__eql, const allocator_type &__a=allocator_type())
- **hash_set** (size_type __n)
- **hash_set** (size_type __n, const hasher &__hf)
- **hash_set** (size_type __n, const hasher &__hf, const key_equal &__eql, const allocator_type &__a=allocator_↩ type())
- iterator **begin** () const
- size_type **bucket_count** () const
- void **clear** ()
- size_type **count** (const key_type &__key) const
- size_type **elems_in_bucket** (size_type __n) const
- bool **empty** () const
- iterator **end** () const
- pair< iterator, iterator > **equal_range** (const key_type &__key) const
- size_type **erase** (const key_type &__key)
- void **erase** (iterator __f, iterator __l)
- void **erase** (iterator __it)
- iterator **find** (const key_type &__key) const
- allocator_type **get_allocator** () const
- hasher **hash_funct** () const
- template<class _InputIterator>
  void **insert** (_InputIterator __f, _InputIterator __l)
- pair< iterator, bool > **insert** (const value_type &__obj)
- pair< iterator, bool > **insert_noresize** (const value_type &__obj)
- key_equal **key_eq** () const
- size_type **max_bucket_count** () const
- size_type **max_size** () const
- void **resize** (size_type __hint)
- size_type **size** () const
- void **swap** (hash_set &__hs)

Friends

- template<class _Val, class _HF, class _EqK, class _Al>
  bool **operator==** (const hash_set< _Val, _HF, _EqK, _Al > &, const hash_set< _Val, _HF, _EqK, _Al > &)

5.516.1 Detailed Description

template<class _Value, class _HashFcn = hash<_Value>, class _EqualKey = equal_to<_Value>, class _Alloc = allocator<_Value>>
class __gnu_cxx::hash_set< _Value, _HashFcn, _EqualKey, _Alloc >

This is an SGI extension.

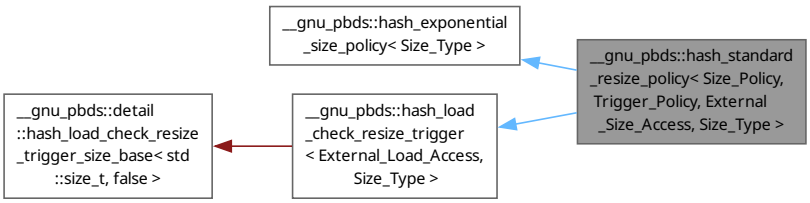
Todo Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

The documentation for this class was generated from the following file:

- hash_set

5.517 __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type > Class Template Reference

`#include <hash_policy.hpp>`
Inheritance diagram for __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >:



Public Types

- enum { external_load_access }
- enum { **external_size_access** }
- typedef Size_Policy **size_policy**
- typedef Size_Type **size_type**
- typedef Trigger_Policy **trigger_policy**

Public Member Functions

- hash_standard_resize_policy ()
- hash_standard_resize_policy (const Size_Policy &r_size_policy)
- hash_standard_resize_policy (const Size_Policy &r_size_policy, const Trigger_Policy &r_trigger_policy)
- size_type get_actual_size () const
- std::pair< float, float > get_loads () const
- Size_Policy & get_size_policy ()

- const Size_Policy & get_size_policy () const
- Trigger_Policy & get_trigger_policy ()
- const Trigger_Policy & get_trigger_policy () const
- void resize (size_type suggested_new_size)
- void set_loads (std::pair< float, float > load_pair)
- void **swap** (hash_exponential_size_policy< Size_Type > &other)
- void **swap** (hash_load_check_resize_trigger &other)
- void **swap** (hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type > &other)

Protected Member Functions

- size_type **get_nearest_larger_size** (size_type size) const
- size_type **get_nearest_smaller_size** (size_type size) const
- size_type get_new_size (size_type size, size_type num_used_e) const
- bool **is_grow_needed** (size_type size, size_type num_entries) const
- bool **is_resize_needed** () const
- void **notify_cleared** ()
- void **notify_erase_search_collision** ()
- void **notify_erase_search_end** ()
- void **notify_erase_search_start** ()
- void **notify_erased** (size_type num_e)
- void **notify_externally_resized** (size_type new_size)
- void **notify_find_search_collision** ()
- void **notify_find_search_end** ()
- void **notify_find_search_start** ()
- void **notify_insert_search_collision** ()
- void **notify_insert_search_end** ()
- void **notify_insert_search_start** ()
- void **notify_inserted** (size_type num_e)
- void **notify_resized** (size_type new_size)

5.517.1 Detailed Description

template<typename Size_Policy = hash_exponential_size_policy<>, typename Trigger_Policy = hash_load_↩ check_resize_trigger<>, bool External_Size_Access = false, typename Size_Type = std::size_t> class __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_↩ Type >

A resize policy which delegates operations to size and trigger policies.

5.517.2 Member Enumeration Documentation

anonymous enum

```
template<bool External_Load_Access = false, typename Size_Type = std::size_t>
anonymous enum  [inherited]
```

Enumerator

external_load_access	Specifies whether the load factor can be accessed externally. The two options have different trade-offs in terms of flexibility, genericity, and encapsulation.

5.517.3 Constructor & Destructor Documentation

hash_standard_resize_policy() [1/3]

```
template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size↩
_Type>
__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size↩
_Type >::hash_standard_resize_policy ()
```

Default constructor.

hash_standard_resize_policy() [2/3]

```
template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size↩
_Type>
__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size↩
_Type >::hash_standard_resize_policy (
            const Size_Policy & r_size_policy)
```

constructor taking some policies r_size_policy will be copied by the Size_Policy object of this object.

hash_standard_resize_policy() [3/3]

```
template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size↩
_Type>
__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size↩
_Type >::hash_standard_resize_policy (
            const Size_Policy & r_size_policy,
            const Trigger_Policy & r_trigger_policy)
```

constructor taking some policies. r_size_policy will be copied by the Size_Policy object of this object. r_trigger_policy will be copied by the Trigger_Policy object of this object.

5.517.4 Member Function Documentation

get_actual_size()

```
template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size↩
_Type>
hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >↩
::size_type __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_↩
Access, Size_Type >::get_actual_size () const  [inline]
```

Returns the actual size of the container.

get_loads()

```
template<bool External_Load_Access, typename Size_Type>
std::pair< float, float > __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size↩
_Type >::get_loads () const  [inline], [inherited]
```

Returns a pair of the minimal and maximal loads, respectively.

get_new_size()

```
template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size↩
_Type>
hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >↩
::size_type __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_↩
Access, Size_Type >::get_new_size (
```

```
            size_type size,
            size_type num_used_e) const  [protected]
```

Queries what the new size should be, when the container is resized naturally. The current __size of the container is size, and the number of used entries within the container is num_used_e.

get_size_policy() [1/2]

```
template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size↩
_Type>
Size_Policy & __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_↩
Size_Access, Size_Type >::get_size_policy ()
```

Access to the Size_Policy object used.

get_size_policy() [2/2]

```
template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size↩
_Type>
const Size_Policy & __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External↩
_Size_Access, Size_Type >::get_size_policy () const
```

Const access to the Size_Policy object used.

get_trigger_policy() [1/2]

```
template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size↩
_Type>
Trigger_Policy & __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_↩
Size_Access, Size_Type >::get_trigger_policy ()
```

Access to the Trigger_Policy object used.

get_trigger_policy() [2/2]

```
template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size↩
_Type>
const Trigger_Policy & __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy,
External_Size_Access, Size_Type >::get_trigger_policy () const
```

Access to the Trigger_Policy object used.

resize()

```
template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size↩
_Type>
void __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access,
Size_Type >::resize (
            size_type suggested_new_size)
```

Resizes the container to suggested_new_size, a suggested size (the actual size will be determined by the Size_Policy object).

set_loads()

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::set_loads (
            std::pair< float, float > load_pair)  [inherited]
```

Sets the loads through a pair of the minimal and maximal loads, respectively.
The documentation for this class was generated from the following file:

- hash_policy.hpp

5.518 std::chrono::hh_mm_ss< _Duration > Class Template Reference

`#include <chrono>`

Public Types

- using **precision**

Public Member Functions

- constexpr **hh_mm_ss** (_Duration __d)
- constexpr chrono::hours **hours** () const noexcept
- constexpr bool **is_negative** () const noexcept
- constexpr chrono::minutes **minutes** () const noexcept
- constexpr **operator precision** () const noexcept
- constexpr chrono::seconds **seconds** () const noexcept
- constexpr precision **subseconds** () const noexcept
- constexpr precision **to_duration** () const noexcept

Static Public Attributes

- static constexpr unsigned **fractional_width**

5.518.1 Detailed Description

template<typename _Duration>
class std::chrono::hh_mm_ss< _Duration >

Utility for splitting a duration into hours, minutes, and seconds
This is a convenience type that provides accessors for the constituent parts (hours, minutes, seconds and subseconds) of a duration.

Since

C++20

The documentation for this class was generated from the following file:

- chrono

5.519 std::locale::id Class Reference

`#include <locale_classes.h>`

Public Member Functions

- id ()
- size_t **_M_id** () const throw ()

Friends

- template<typename _Facet>
  const _Facet * **__try_use_facet** (const locale &__loc)
- template<typename _Facet>
  bool has_facet (const locale &__loc) throw ()
- class **locale**
- class **locale::_Impl**
- template<typename _Facet>
  const _Facet & use_facet (const locale &__loc)

5.519.1 Detailed Description

Facet ID class.
The ID class provides facets with an index used to identify them. Every facet class must define a public static member locale::id, or be derived from a facet that provides this member, otherwise the facet cannot be used in a locale. The locale::id ensures that each class type gets a unique identifier.

5.519.2 Constructor & Destructor Documentation

id()

```
std::locale::id::id ()  [inline]
```
Constructor.

5.519.3 Friends And Related Symbol Documentation

has_facet

```
template<typename _Facet>
bool has_facet (
            const locale & __loc) throw ( )   [friend]
```
Test for the presence of a facet.
has_facet tests the locale argument for the presence of the facet type provided as the template parameter. Facets derived from the facet parameter will also return true.

Template Parameters

_Facet	The facet type to test the presence of.

Parameters

__loc	The locale to test.

Returns

true if __loc contains a facet of type _Facet, else false.

use_facet

```
template<typename _Facet>
const _Facet & use_facet (
            const locale & __loc)  [friend]
```
Return a facet.
use_facet looks for and returns a reference to a facet of type Facet where Facet is the template parameter. If has_facet(locale) is true, there is a suitable facet to return. It throws std::bad_cast if the locale doesn't contain a facet of type Facet.

Template Parameters

_Facet	The facet type to access.

Parameters

__loc	The locale to use.

Returns

Reference to facet of type Facet.

Exceptions

std::bad_cast	if __loc doesn't contain a facet of type _Facet.

The documentation for this class was generated from the following file:

- locale_classes.h

5.520 std::thread::id Class Reference

`#include <thread>`

Public Member Functions

- **id** (native_handle_type __id)

Friends

- struct **hash**< **id** >
- template<class _CharT, class _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__out, id __id)
- strong_ordering **operator**<**=**> (id __x, id __y) noexcept
- bool **operator==** (id __x, id __y) noexcept
- class **thread**

Related Symbols

(Note that these are not member symbols.)

- bool **operator==** (thread::id __x, thread::id __y) noexcept
- strong_ordering **operator**<**=**> (thread::id __x, thread::id __y) noexcept
- template<class _CharT, class _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__out, thread::id __id)

5.520.1 Detailed Description

A std::thread::id is a unique identifier for a thread.

Since

C++11

The documentation for this class was generated from the following files:

- std_thread.h
- thread

5.521 std::identity Struct Reference

`#include <ranges_cmp.h>`

Public Types

- using **is_transparent**

Public Member Functions

- template<typename _Tp>
  constexpr _Tp && **operator()** (_Tp &&__t) const noexcept

5.521.1 Detailed Description

[func.identity] The identity function.
The documentation for this struct was generated from the following file:

- ranges_cmp.h

5.522 std::experimental::fundamentals_v1::in_place_t Struct Reference

`#include <optional>`

5.522.1 Detailed Description

Tag type for in-place construction.
The documentation for this struct was generated from the following file:

- experimental/optional

5.523 std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > Class Template Reference

`#include <random>`

Public Types

- typedef _UIntType result_type

Public Member Functions

- independent_bits_engine ()
- independent_bits_engine (_RandomNumberEngine &&__rng)
- template<typename _Sseq, typename = _If_seed_seq<_Sseq>>
  independent_bits_engine (_Sseq &__q)
- independent_bits_engine (const _RandomNumberEngine &__rng)
- independent_bits_engine (result_type __s)
- const _RandomNumberEngine & base () const noexcept
- void discard (unsigned long long __z)
- result_type operator() ()
- void seed ()
- template<typename _Sseq>
  _If_seed_seq< _Sseq > seed (_Sseq &__q)
- void seed (result_type __s)

Static Public Member Functions

- static constexpr result_type max ()
- static constexpr result_type min ()

Friends

- bool operator== (const independent_bits_engine &__lhs, const independent_bits_engine &__rhs)
- template<typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__x)

5.523.1 Detailed Description

template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
class std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >

Produces random numbers by combining random numbers from some base engine to produce random numbers with a specified number of bits __w.

Since

C++11

5.523.2 Member Typedef Documentation

result_type

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
typedef _UIntType std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::result_↩
type
```

The type of the generated random value.

5.523.3 Constructor & Destructor Documentation

independent_bits_engine() [1/5]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::independent_bits_engine ()
[inline]
```

Constructs a default independent_bits_engine engine.
The underlying engine is default constructed as well.
Referenced by operator==.

independent_bits_engine() [2/5]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::independent_bits_engine (
            const _RandomNumberEngine & __rng)  [inline], [explicit]
```

Copy constructs a independent_bits_engine engine.
Copies an existing base class random number generator.

Parameters

__rng	An existing (base class) engine object.

independent_bits_engine() [3/5]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::independent_bits_engine (
            _RandomNumberEngine && __rng)  [inline], [explicit]
```

Move constructs a independent_bits_engine engine.
Copies an existing base class random number generator.

Parameters

__rng	An existing (base class) engine object.

References std::move().

independent_bits_engine() [4/5]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::independent_bits_engine (
            result_type __s)  [inline], [explicit]
```

Seed constructs a independent_bits_engine engine.
Constructs the underlying generator engine seeded with __s.

Parameters

_↩ _s	A seed value for the base class engine.

independent_bits_engine() [5/5]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
template<typename _Sseq, typename = _If_seed_seq<_Sseq>>
std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::independent_bits_engine (
            _Sseq & __q)  [inline], [explicit]
```

Generator construct a independent_bits_engine engine.

Parameters

_↩ _q	A seed sequence.

5.523.4 Member Function Documentation

base()

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
const _RandomNumberEngine & std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >↩
::base () const  [inline], [noexcept]
```

Gets a const reference to the underlying generator engine object.

discard()

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
void std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::discard (
            unsigned long long __z)  [inline]
```

Discard a sequence of random numbers.

max()

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
static constexpr result_type std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType
>::max ()  [inline], [static], [constexpr]
```

Gets the maximum value in the generated random number range.

min()

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
static constexpr result_type std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType
>::min ()  [inline], [static], [constexpr]
```

Gets the minimum value in the generated random number range.

operator()()

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::result_type std::independent_bits_engine<
_RandomNumberEngine, __w, _UIntType >::operator() ()
```

Gets the next value in the generated random number sequence.

References std::__lg(), std::__numeric_limits_base::digits, and std::numeric_limits< _Tp >::max().

seed() [1/3]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
void std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::seed ()  [inline]
```

Reseeds the independent_bits_engine object with the default seed for the underlying base class generator engine.

seed() [2/3]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
template<typename _Sseq>
_If_seed_seq< _Sseq > std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::seed
(
              _Sseq & __q)  [inline]
```

Reseeds the independent_bits_engine object with the given seed sequence.

Parameters

_↩ _q	A seed generator function.

seed() [3/3]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
void std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::seed (
            result_type __s)  [inline]
```

Reseeds the independent_bits_engine object with the default seed for the underlying base class generator engine.

5.523.5 Friends And Related Symbol Documentation

operator==

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
bool operator== (
```

```
            const independent_bits_engine< _RandomNumberEngine, __w, _UIntType > & __lhs,
            const independent_bits_engine< _RandomNumberEngine, __w, _UIntType > & __rhs)  [friend]
```

Compares two independent_bits_engine random number generator objects of the same type for equality.

Parameters

__lhs	A independent_bits_engine random number generator object.
__rhs	Another independent_bits_engine random number generator object.

Returns

true if the infinite sequences of generated values would be equal, false otherwise.

References independent_bits_engine().

operator>>

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
template<typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > & __x)  [friend]
```

Extracts the current state of a % subtract_with_carry_engine random number generator engine __x from the input stream __is.

Parameters

__is	An input stream.
__x	A independent_bits_engine random number generator engine.

Returns

The input stream with the state of __x extracted or in an error state.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

5.524 std::indirect_array< _Tp > Class Template Reference

```
#include <indirect_array.h>
```

Public Types

- typedef _Tp **value_type**

Public Member Functions

- indirect_array (const indirect_array &)
- template<class _Dom>
  void **operator%=** (const _Expr< _Dom, _Tp > &) const
- void operator%= (const valarray< _Tp > &) const

- template<class _Dom>
  void **operator&=** (const _Expr< _Dom, _Tp > &) const
- void operator&= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator*=** (const _Expr< _Dom, _Tp > &) const
- void operator*= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator+=** (const _Expr< _Dom, _Tp > &) const
- void operator+= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator-=** (const _Expr< _Dom, _Tp > &) const
- void operator-= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator/=** (const _Expr< _Dom, _Tp > &) const
- void operator/= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator<<=** (const _Expr< _Dom, _Tp > &) const
- void operator<<= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator=** (const _Expr< _Dom, _Tp > &) const
- void operator= (const _Tp &) const
- indirect_array & operator= (const indirect_array &)
- void operator= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator>>=** (const _Expr< _Dom, _Tp > &) const
- void operator>>= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator^=** (const _Expr< _Dom, _Tp > &) const
- void operator^= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator|=** (const _Expr< _Dom, _Tp > &) const
- void operator|= (const valarray< _Tp > &) const

Friends

- class **gslice_array**< **_Tp** >
- class **valarray**< **_Tp** >

5.524.1 Detailed Description

template<**class _Tp**>
class std::indirect_array< **_Tp** >

Reference to arbitrary subset of an array.
An indirect_array is a reference to the actual elements of an array specified by an ordered array of indices. The way to get an indirect_array is to call operator[](valarray<size_t>) on a valarray. The returned indirect_array then permits carrying operations out on the referenced subset of elements in the original valarray.
For example, if an indirect_array is obtained using the array (4,2,0) as an argument, and then assigned to an array containing (1,2,3), then the underlying array will have array[0]==3, array[2]==2, and array[4]==1.

Parameters

Tp	Element type.

5.524.2 Member Function Documentation

operator%=()

```
template<class _Tp>
void std::indirect_array< _Tp >::operator%= (
            const valarray< _Tp > & ) const
```

Modulo slice elements by corresponding elements of *v*.

operator&=()

```
template<class _Tp>
void std::indirect_array< _Tp >::operator&= (
            const valarray< _Tp > & ) const
```

Logical and slice elements with corresponding elements of *v*.

operator*=()

```
template<class _Tp>
void std::indirect_array< _Tp >::operator*= (
            const valarray< _Tp > & ) const
```

Multiply slice elements by corresponding elements of *v*.

operator+=()

```
template<class _Tp>
void std::indirect_array< _Tp >::operator+= (
            const valarray< _Tp > & ) const
```

Add corresponding elements of *v* to slice elements.

operator-=()

```
template<class _Tp>
void std::indirect_array< _Tp >::operator-= (
            const valarray< _Tp > & ) const
```

Subtract corresponding elements of *v* from slice elements.

operator/=()

```
template<class _Tp>
void std::indirect_array< _Tp >::operator/= (
            const valarray< _Tp > & ) const
```

Divide slice elements by corresponding elements of *v*.

operator<<=()

```
template<class _Tp>
void std::indirect_array< _Tp >::operator<<= (
            const valarray< _Tp > & ) const
```

Left shift slice elements by corresponding elements of *v*.

operator>>=()

```
template<class _Tp>
void std::indirect_array< _Tp >::operator>>= (
            const valarray< _Tp > & ) const
```

Right shift slice elements by corresponding elements of *v*.
References indirect_array().

operator^=()

```
template<class _Tp>
void std::indirect_array< _Tp >::operator^= (
            const valarray< _Tp > & ) const
```

Logical xor slice elements with corresponding elements of *v*.

operator"" |=()

```
template<class _Tp>
void std::indirect_array< _Tp >::operator|= (
            const valarray< _Tp > & ) const
```

Logical or slice elements with corresponding elements of *v*.
The documentation for this class was generated from the following files:

- valarray
- indirect_array.h

5.525 std::initializer_list< _E > Class Template Reference

`#include <initializer_list>`

Public Types

- typedef const _E * **const_iterator**
- typedef const _E & **const_reference**
- typedef const _E * **iterator**
- typedef const _E & **reference**
- typedef size_t **size_type**
- typedef _E **value_type**

Public Member Functions

- constexpr const_iterator **begin** () const noexcept
- constexpr const_iterator **end** () const noexcept
- constexpr size_type **size** () const noexcept

Related Symbols

(Note that these are not member symbols.)

- template<class _Tp>
  constexpr const _Tp * begin (initializer_list< _Tp > __ils) noexcept
- template<class _Tp>
  constexpr const _Tp * end (initializer_list< _Tp > __ils) noexcept

5.525.1 Detailed Description

template<class _E>
class std::initializer_list< _E >

initializer_list

5.525.2 Friends And Related Symbol Documentation

begin()

```
template<class _Tp>
const _Tp * begin (
            initializer_list< _Tp > __ils)  [related]
```

Return an iterator pointing to the first element of the initializer_list.

Parameters

_↩ _ils	Initializer list.

end()

```
template<class _Tp>
const _Tp * end (
            initializer_list< _Tp > __ils)  [related]
```

Return an iterator pointing to one past the last element of the initializer_list.

Parameters

_↩ _ils	Initializer list.

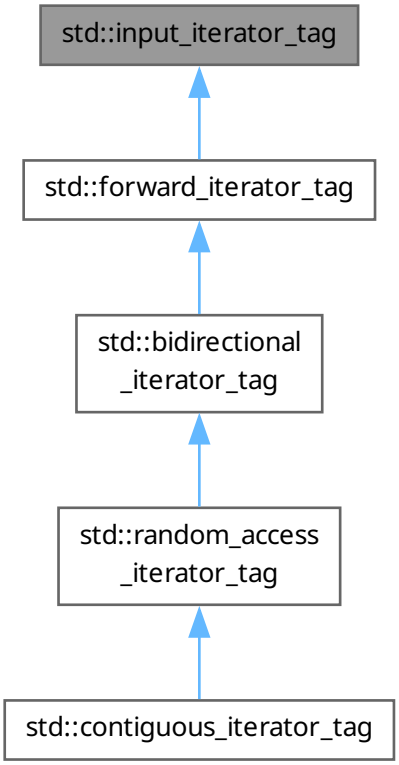
The documentation for this class was generated from the following file:

- initializer_list

5.526 std::input_iterator_tag Struct Reference

```
#include <stl_iterator_base_types.h>
```

Inheritance diagram for std::input_iterator_tag:



5.526.1 Detailed Description

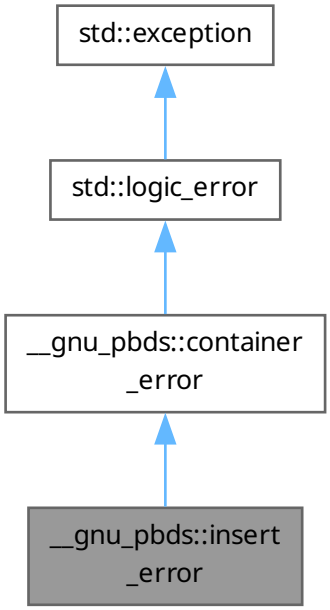
Marking input iterators.
The documentation for this struct was generated from the following file:

- stl_iterator_base_types.h

5.527 __gnu_pbds::insert_error Struct Reference

`#include <exception.hpp>`

Inheritance diagram for __gnu_pbds::insert_error:



Public Member Functions

- virtual const char ∗ what () const noexcept

5.527.1 Detailed Description

An entry cannot be inserted into a container object for logical reasons (not, e.g., if memory is unabvailable, in which case the allocator_type's exception will be thrown).

5.527.2 Member Function Documentation

what()

```
virtual const char * std::logic_error::what () const  [virtual], [noexcept], [inherited]
```

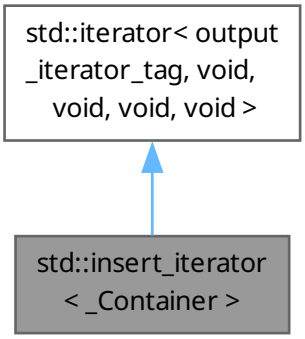
Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).
Reimplemented from std::exception.
Reimplemented in std::future_error.
The documentation for this struct was generated from the following file:

- exception.hpp

5.528 std::insert_iterator< _Container > Class Template Reference

```
#include <stl_iterator.h>
```

Inheritance diagram for std::insert_iterator< _Container >:



Public Types

- typedef _Container container_type
- using **difference_type**
- typedef output_iterator_tag iterator_category
- typedef void pointer
- typedef void reference
- typedef void value_type

Public Member Functions

- constexpr insert_iterator (_Container &__x, _Iter __i)
- constexpr insert_iterator & operator∗ ()
- constexpr insert_iterator & operator++ ()
- constexpr insert_iterator & operator++ (int)
- constexpr insert_iterator & operator= (const typename _Container::value_type &__value)
- constexpr insert_iterator & **operator=** (typename _Container::value_type &&__value)

Protected Attributes

- _Container ∗ **container**
- _Iter **iter**

5.528.1 Detailed Description

template<**typename _Container**>
class std::insert_iterator< **_Container** >

Turns assignment into insertion.
These are output iterators, constructed from a container-of-T. Assigning a T to the iterator inserts it in the container at the iterator's position, rather than overwriting the value at that position.
(Sequences will actually insert a *copy* of the value before the iterator's position.)
Tip: Using the inserter function to create these iterators can save typing.

5.528.2 Member Typedef Documentation

container_type

```
template<typename _Container>
typedef _Container std::insert_iterator< _Container >::container_type
```

A nested typedef for the type of whatever container you used.

iterator_category

```
typedef output_iterator_tag std::iterator< output_iterator_tag, void, void, void, void >::iterator↩
_category  [inherited]
```

One of the tag types.

pointer

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::pointer  [inherited]
```

This type represents a pointer-to-value_type.

reference

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::reference  [inherited]
```

This type represents a reference-to-value_type.

value_type

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::value_type  [inherited]
```

The type "pointed to" by the iterator.

5.528.3 Constructor & Destructor Documentation

insert_iterator()

```
template<typename _Container>
std::insert_iterator< _Container >::insert_iterator (
            _Container & __x,
            _Iter __i)  [inline], [constexpr]
```

The only way to create this iterator is with a container and an initial position (a normal iterator into the container).
References std::__addressof().

5.528.4 Member Function Documentation

operator∗()

```
template<typename _Container>
insert_iterator & std::insert_iterator< _Container >::operator* ()  [inline], [nodiscard], [constexpr]
```

Simply returns ∗this.

operator++() [1/2]

```
template<typename _Container>
insert_iterator & std::insert_iterator< _Container >::operator++ ()  [inline], [constexpr]
```

Simply returns ∗this. (This iterator does not *move*.)

operator++() [2/2]

```
template<typename _Container>
insert_iterator & std::insert_iterator< _Container >::operator++ (
            int )  [inline], [constexpr]
```

Simply returns *this. (This iterator does not *move*.)

operator=()

```
template<typename _Container>
insert_iterator & std::insert_iterator< _Container >::operator= (
            const typename _Container::value_type & __value)  [inline], [constexpr]
```

Parameters

__value	An instance of whatever type container_type::const_reference is; presumably a reference-to-const T for container<T>.

Returns

This iterator, for chained operations.

This kind of iterator maintains its own position in the container. Assigning a value to the iterator will insert the value into the container at the place before the iterator.
The position is maintained such that subsequent assignments will insert values immediately after one another. For example,

```
// vector v contains A and Z

insert_iterator i (v, ++v.begin());
i = 1;
i = 2;
i = 3;

// vector v contains A, 1, 2, 3, and Z
```

The documentation for this class was generated from the following file:

- bits/stl_iterator.h

5.529 std::integer_sequence< _Tp, _Idx > Struct Template Reference

```
#include <utility.h>
```

Public Types

- typedef _Tp **value_type**

Static Public Member Functions

- static constexpr size_t **size** () noexcept

5.529.1 Detailed Description

template<typename _Tp, _Tp... _Idx>
struct std::integer_sequence< _Tp, _Idx >

Class template integer_sequence.
The documentation for this struct was generated from the following file:

- utility.h

5.530 std::integral_constant< _Tp, __v > Struct Template Reference

`#include <type_traits>`

Public Types

- using **type**
- using **value_type**

Public Member Functions

- constexpr **operator value_type** () const noexcept

Static Public Attributes

- static constexpr _Tp **value**

5.530.1 Detailed Description

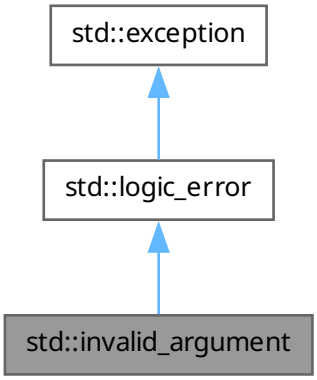
template<typename _Tp, _Tp __v>
struct std::integral_constant< _Tp, __v >

integral_constant
The documentation for this struct was generated from the following file:

- type_traits

5.531 std::invalid_argument Class Reference

`#include <stdexcept>`
Inheritance diagram for std::invalid_argument:



Public Member Functions

- **invalid_argument** (const char *)
- **invalid_argument** (const invalid_argument &)=default

- **invalid_argument** (const string &__arg)
- **invalid_argument** (invalid_argument &&)=default
- invalid_argument & **operator=** (const invalid_argument &)=default
- invalid_argument & **operator=** (invalid_argument &&)=default
- virtual const char * what () const noexcept

5.531.1 Detailed Description

Thrown to report invalid arguments to functions.

5.531.2 Member Function Documentation

what()

```
virtual const char * std::logic_error::what () const  [virtual], [noexcept], [inherited]
```

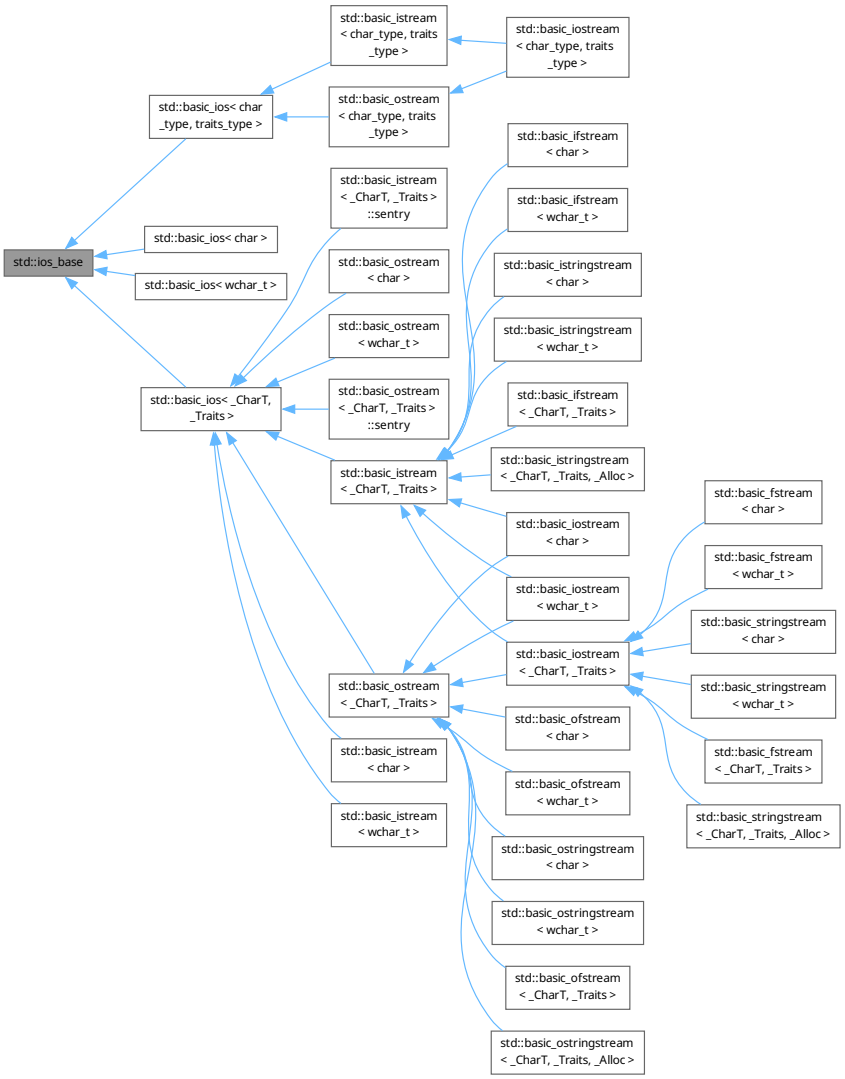
Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).
Reimplemented from std::exception.
Reimplemented in std::future_error.
The documentation for this class was generated from the following file:

- stdexcept

5.532 std::ios_base Class Reference

```
#include <ios_base.h>
```

Inheritance diagram for std::ios_base:



Public Types

- enum event { **erase_event** , **imbue_event** , **copyfmt_event** }
- typedef void(∗ event_callback) (event __e, ios_base &__b, int __i)
- typedef _Ios_Iostate iostate
- typedef _Ios_Openmode openmode
- typedef _Ios_Seekdir seekdir

Public Member Functions

- **ios_base** (const ios_base &)=delete
- virtual ∼ios_base ()
- const locale & _M_getloc () const
- fmtflags flags () const

- fmtflags flags (fmtflags __fmtfl)
- locale getloc () const
- locale imbue (const locale &__loc) throw ()
- long & iword (int __ix)
- ios_base & **operator=** (const ios_base &)=delete
- streamsize precision () const
- streamsize precision (streamsize __prec)
- void *& pword (int __ix)
- void register_callback (event_callback __fn, int __index)
- fmtflags setf (fmtflags __fmtfl)
- fmtflags setf (fmtflags __fmtfl, fmtflags __mask)
- void unsetf (fmtflags __mask)
- streamsize width () const
- streamsize width (streamsize __wide)

Static Public Member Functions

- static bool sync_with_stdio (bool __sync=true)
- static int xalloc () throw ()

Public Attributes

- class __attribute((__abi_tag__("cxx11"))) failure typedef _Ios_Fmtflags fmtflags

Static Public Attributes

- static const openmode **__noreplace**
- static const fmtflags adjustfield
- static const openmode app
- static const openmode ate
- static const iostate badbit
- static const fmtflags basefield
- static const seekdir beg
- static const openmode binary
- static const fmtflags boolalpha
- static const seekdir cur
- static const fmtflags dec
- static const seekdir end
- static const iostate eofbit
- static const iostate failbit
- static const fmtflags fixed
- static const fmtflags floatfield
- static const iostate goodbit
- static const fmtflags hex
- static const openmode in
- static const fmtflags internal
- static const fmtflags left
- static const fmtflags oct
- static const openmode out
- static const fmtflags right
- static const fmtflags scientific
- static const fmtflags showbase

- static const fmtflags showpoint
- static const fmtflags showpos
- static const fmtflags skipws
- static const openmode trunc
- static const fmtflags unitbuf
- static const fmtflags uppercase

Protected Types

- enum { **_S_local_word_size** }

Protected Member Functions

- void **_M_call_callbacks** (event __ev) throw ()
- void **_M_dispose_callbacks** (void) throw ()
- _Words & **_M_grow_words** (int __index, bool __iword)
- void **_M_init** () throw ()
- void **_M_move** (ios_base &) noexcept
- void **_M_swap** (ios_base &__rhs) noexcept

Protected Attributes

- _Callback_list * **_M_callbacks**
- iostate **_M_exception**
- fmtflags **_M_flags**
- locale **_M_ios_locale**
- _Words **_M_local_word** [_S_local_word_size]
- streamsize **_M_precision**
- iostate **_M_streambuf_state**
- streamsize **_M_width**
- _Words * **_M_word**
- int **_M_word_size**
- _Words **_M_word_zero**

5.532.1 Detailed Description

The base of the I/O class hierarchy.
This class defines everything that can be defined about I/O that does not depend on the type of characters being input or output. Most people will only see `ios_base` when they need to specify the full name of the various I/O flags (e.g., the openmodes).

5.532.2 Member Typedef Documentation

event_callback

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i)
```

The type of an event callback function.

Parameters

_↩ _e	One of the members of the event enum.
_↩ _b	Reference to the ios_base object.

_↩ _*i*	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several ios_base and basic_ios functions, specifically imbue(), copyfmt(), and ~ios().

iostate

`typedef _Ios_Iostate std::ios_base::iostate`
This is a bitmask type.
_Ios_Iostate is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit
- eofbit
- failbit
- goodbit

openmode

`typedef _Ios_Openmode std::ios_base::openmode`
This is a bitmask type.
_Ios_Openmode is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type openmode are:

- app
- ate
- binary
- in
- out
- trunc

seekdir

`typedef _Ios_Seekdir std::ios_base::seekdir`
This is an enumerated type.
_Ios_Seekdir is implementation-defined. Defined values of type seekdir are:

- beg
- cur, equivalent to SEEK_CUR in the C standard library.
- end, equivalent to SEEK_END in the C standard library.

5.532.3 Member Enumeration Documentation

event

`enum std::ios_base::event`
The set of events that may be passed to an event callback.
erase_event is used during ~ios() and copyfmt(). imbue_event is used during imbue(). copyfmt_event is used during copyfmt().

5.532.4 Constructor & Destructor Documentation

~ios_base()

```
virtual std::ios_base::~ios_base ()  [virtual]
```

Invokes each callback with erase_event. Destroys local storage.
Note that the ios_base object for the standard streams never gets destroyed. As a result, any callbacks registered with the standard streams will not get invoked with erase_event (unless copyfmt is used).

5.532.5 Member Function Documentation

_M_getloc()

```
const locale & std::ios_base::_M_getloc () const  [inline]
```

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.
Referenced by std::money_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIte std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::num_put< _CharT, _OutIter std::time_put< _CharT, _OutIter >::do_put(), std::time_get< _CharT, _InIter >::get(), and std::time_put< _CharT, _OutIter >::put().

flags() [1/2]

```
fmtflags std::ios_base::flags () const  [inline], [nodiscard]
```

Access to format flags.

Returns

The format control flags for both input and output.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter > std::num_put< _CharT, _OutIter >::do_put(), std::num_put< _CharT, _OutIter >::do_put(), std::chrono::operator<<(), std::operator<<(), std::__detail::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), and std::operator>>().

flags() [2/2]

```
fmtflags std::ios_base::flags (
            fmtflags __fmtfl)  [inline]
```

Setting new format flags all at once.

Parameters

__fmtfl	The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *__fmtfl*.
References fmtflags.

getloc()

```
locale std::ios_base::getloc () const  [inline], [nodiscard]
```

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::money_put< _CharT, _OutIter >::do_put(), std::basic_ios< _CharT, _Tr
std::chrono::operator<<(), std::operator>>(), std::operator>>(), and std::ws().

imbue()

```
locale std::ios_base::imbue (
            const locale & __loc) throw ( )
```

Setting a new locale.

Parameters

__loc	The new locale.

Returns

The previous locale.

Sets the new locale for this stream, and then invokes each callback with imbue_event.
Referenced by std::basic_ios< _CharT, _Traits >::imbue().

iword()

```
long & std::ios_base::iword (
            int __ix)  [inline]
```

Access to integer array.

Parameters

__ix	Index into the array.

Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.
References iword().
Referenced by iword().

precision() [1/2]

```
streamsize std::ios_base::precision () const  [inline], [nodiscard]
```

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::chrono::operator<<().

precision() [2/2]

```
streamsize std::ios_base::precision (
            streamsize __prec)  [inline]
```

Changing flags.

Parameters

__prec	The new precision value.

Returns

The previous value of precision().

pword()

```
void *& std::ios_base::pword (
            int __ix)  [inline]
```

Access to void pointer array.

Parameters

_↩ _ix	Index into the array.

Returns

A reference to a void* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

register_callback()

```
void std::ios_base::register_callback (
            event_callback __fn,
            int __index)
```

Add the callback __fn with parameter __index.

Parameters

__fn	The function to add.

__index	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.
References fmtflags.

setf() [1/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl)  [inline]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.
References fmtflags.
Referenced by std::__detail::operator>>().

setf() [2/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl,
            fmtflags __mask)  [inline]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.
__mask	The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* `&` *mask*. An example mask is `ios_base::adjustfield`.
References fmtflags.

sync_with_stdio()

```
static bool std::ios_base::sync_with_stdio (
            bool __sync = true)  [static]
```

Interaction with the standard C I/O objects.

Parameters

__sync	Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary

unsetf()

```
void std::ios_base::unsetf (
            fmtflags __mask)  [inline]
```

Clearing format flags.

Parameters

__mask	The flags to unset.

This function clears *__mask* in the format flags.
References fmtflags.

width() [1/2]

```
streamsize std::ios_base::width () const  [inline], [nodiscard]
```

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_put< _CharT, _OutIter >::do_put(), std::operator>>(), and std::operator>>().

width() [2/2]

```
streamsize std::ios_base::width (
            streamsize __wide)  [inline]
```

Changing flags.

Parameters

__wide	The new width value.

Returns

The previous value of width().

xalloc()

```
static int std::ios_base::xalloc () throw ( )  [static]
```

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the iword and pword functions. The expectation is that an application calls xalloc in order to obtain an index in the iword and pword arrays that can be used without fear of conflict.
The implementation maintains a static variable that is incremented and returned on each invocation. xalloc is guaranteed to return an index that is safe to use in the iword and pword arrays.

5.532.6 Member Data Documentation

adjustfield

```
const fmtflags std::ios_base::adjustfield  [static]
```

A mask of left|right|internal. Useful for the 2-arg form of `setf`.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::internal(), std::left(), and std::right().

app

```
const openmode std::ios_base::app  [static]
```

Seek to end before each write.
Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

ate

```
const openmode std::ios_base::ate  [static]
```

Open and seek to end immediately after opening.
Referenced by std::basic_filebuf< _CharT, _Traits >::open().

badbit

```
const iostate std::ios_base::badbit  [static]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_istream< char_type, traits_type >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::basic_ios< _CharT, _Traits >::clear(), std::basic_ostream< _CharT, _Trai std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_ostream< _CharT, _Traits >::opera std::basic_ostream< char_type, traits_type >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >: std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< char_type, traits_ty std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_istream< char_type, traits_type and std::ws().

basefield

```
const fmtflags std::ios_base::basefield  [static]
```

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.
Referenced by std::dec(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::hex(), and std::oct().

beg

```
const seekdir std::ios_base::beg  [static]
```

Request a seek relative to the beginning of the stream.
Referenced by std::basic_filebuf< _CharT, _Traits >::seekpos().

binary

```
const openmode std::ios_base::binary [static]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary.

Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

boolalpha

```
const fmtflags std::ios_base::boolalpha [static]
```

Insert/extract `bool` in alphabetic rather than numeric format.

Referenced by std::boolalpha(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::noboolalpha().

cur

```
const seekdir std::ios_base::cur [static]
```

Request a seek relative to the current position within the sequence.

Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

dec

```
const fmtflags std::ios_base::dec [static]
```

Converts integer input or generates integer output in decimal base.

Referenced by std::dec().

end

```
const seekdir std::ios_base::end [static]
```

Request a seek relative to the current end of the sequence.

Referenced by std::basic_filebuf< _CharT, _Traits >::open(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff().

eofbit

```
const iostate std::ios_base::eofbit [static]
```

Indicates that an input operation reached the end of an input sequence.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >
std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(),
std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter
std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::ge
std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< char_type, traits_type >::g
std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::putback(),
std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< char_type, traits_type >::read(), std::basic_istream< _CharT, _Traits
std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< char_type, traits_ty
std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

failbit

```
const iostate std::ios_base::failbit [static]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_weekday(),

std::time_get< _CharT, _InIter >::do_get_year(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::ge
std::basic_istream< char_type, traits_type >::get(), std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::oper
std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(),
std::basic_istream< _CharT, _Traits >::read(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::s
std::basic_ostream< _CharT, _Traits >::seekp(), and std::basic_ostream< _CharT, _Traits >::seekp().

fixed

```
const fmtflags std::ios_base::fixed  [static]
```

Generate floating-point output in fixed-point notation.
Referenced by std::fixed(), and std::hexfloat().

floatfield

```
const fmtflags std::ios_base::floatfield  [static]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.
Referenced by std::defaultfloat(), std::fixed(), std::hexfloat(), and std::scientific().

goodbit

```
const iostate std::ios_base::goodbit  [static]
```

Indicates all is well.
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::num_get< _CharT, _InIter >::do_get(),
std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::d
std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >:
std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(),
std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore()
std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_ostream< _CharT, _Traits >
std::basic_ostream< char_type, traits_type >::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(),
std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(),
std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::
std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits
std::basic_istream< char_type, traits_type >::seekg(), std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Trait
and std::ws().

hex

```
const fmtflags std::ios_base::hex  [static]
```

Converts integer input or generates integer output in hexadecimal base.
Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and
std::hex().

in

```
const openmode std::ios_base::in  [static]
```

Open for input. Default for `ifstream` and fstream.
Referenced by std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(),
std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_stringbuf< _CharT, _Traits, _
std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_filebuf< _CharT, _Traits >::showmanyc(),
std::basic_istream< _CharT, _Traits >::tellg(), std::basic_filebuf< _CharT, _Traits >::underflow(), std::basic_stringbuf< _CharT, _Traits, _
and std::basic_filebuf< _CharT, _Traits >::xsgetn().

internal

```
const fmtflags std::ios_base::internal  [static]
```

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.
Referenced by std::internal().

left

`const fmtflags std::ios_base::left [static]`
Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)
Referenced by std::num_put< _CharT, _OutIter >::do_put(), and std::left().

oct

`const fmtflags std::ios_base::oct [static]`
Converts integer input or generates integer output in octal base.
Referenced by std::oct().

out

`const openmode std::ios_base::out [static]`
Open for output. Default for `ofstream` and fstream.
Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_stringbuf< _CharT, _Traits, std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

right

`const fmtflags std::ios_base::right [static]`
Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)
Referenced by std::right().

scientific

`const fmtflags std::ios_base::scientific [static]`
Generates floating-point output in scientific notation.
Referenced by std::hexfloat(), and std::scientific().

showbase

`const fmtflags std::ios_base::showbase [static]`
Generates a prefix indicating the numeric base of generated integer output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::noshowbase(), and std::showbase().

showpoint

`const fmtflags std::ios_base::showpoint [static]`
Generates a decimal-point character unconditionally in generated floating-point output.
Referenced by std::noshowpoint(), and std::showpoint().

showpos

`const fmtflags std::ios_base::showpos [static]`
Generates a + sign in non-negative generated numeric output.
Referenced by std::noshowpos(), and std::showpos().

skipws

```
const fmtflags std::ios_base::skipws  [static]
```
Skips leading white space before certain input operations.
Referenced by std::noskipws(), std::__detail::operator>>(), and std::skipws().

trunc

```
const openmode std::ios_base::trunc  [static]
```
Truncate an existing stream when opening. Default for `ofstream`.

unitbuf

```
const fmtflags std::ios_base::unitbuf  [static]
```
Flushes output after each output operation.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::nounitbuf(), and std::unitbuf().

uppercase

```
const fmtflags std::ios_base::uppercase  [static]
```
Replaces certain lowercase letters with their uppercase equivalents in generated output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::nouppercase(), and std::uppercase().
The documentation for this class was generated from the following file:

- ios_base.h

5.533 std::is_abstract< _Tp > Struct Template Reference

`#include <type_traits>`

5.533.1 Detailed Description

template<typename _Tp>
struct std::is_abstract< _Tp >

is_abstract
The documentation for this struct was generated from the following file:

- type_traits

5.534 std::is_arithmetic< _Tp > Struct Template Reference

`#include <type_traits>`

5.534.1 Detailed Description

template<typename _Tp>
struct std::is_arithmetic< _Tp >

is_arithmetic
The documentation for this struct was generated from the following file:

- type_traits

5.535 std::is_array< _Tp > Struct Template Reference

`#include <type_traits>`

5.535.1 Detailed Description

template<typename _Tp>
struct std::is_array< _Tp >

is_array
The documentation for this struct was generated from the following file:

- type_traits

5.536 std::is_assignable< _Tp, _Up > Struct Template Reference

`#include <type_traits>`

5.536.1 Detailed Description

template<typename _Tp, typename _Up>
struct std::is_assignable< _Tp, _Up >

is_assignable
The documentation for this struct was generated from the following file:

- type_traits

5.537 std::is_base_of< _Base, _Derived > Struct Template Reference

`#include <type_traits>`

5.537.1 Detailed Description

template<typename _Base, typename _Derived>
struct std::is_base_of< _Base, _Derived >

is_base_of
The documentation for this struct was generated from the following file:

- type_traits

5.538 std::is_bind_expression< _Tp > Struct Template Reference

`#include <functional>`

5.538.1 Detailed Description

template<typename _Tp>
struct std::is_bind_expression< _Tp >

Trait that identifies a bind expression.
Determines if the given type `_Tp` is a function object that should be treated as a subexpression when evaluating calls to function objects returned by `std::bind`.
C++11 [func.bind.isbind].

Since

C++11

The documentation for this struct was generated from the following file:

- functional

5.539 std::is_bind_expression< _Bind< _Signature > > Struct Template Reference

`#include <functional>`

5.539.1 Detailed Description

template<typename _Signature>
struct std::is_bind_expression< _Bind< _Signature > >

Class template _Bind is always a bind expression.
The documentation for this struct was generated from the following file:

- functional

5.540 std::is_bind_expression< _Bind_result< _Result, _Signature > > Struct Template Reference

`#include <functional>`

5.540.1 Detailed Description

template<typename _Result, typename _Signature>
struct std::is_bind_expression< _Bind_result< _Result, _Signature > >

Class template _Bind_result is always a bind expression.
The documentation for this struct was generated from the following file:

- functional

5.541 std::is_bind_expression< const _Bind< _Signature > > Struct Template Reference

`#include <functional>`

5.541.1 Detailed Description

template<typename _Signature>
struct std::is_bind_expression< const _Bind< _Signature > >

Class template _Bind is always a bind expression.
The documentation for this struct was generated from the following file:

- functional

5.542 std::is_bind_expression< const _Bind_result< _Result, _Signature > > Struct Template Reference

`#include <functional>`

5.542.1 Detailed Description

template<typename _Result, typename _Signature>
struct std::is_bind_expression< const _Bind_result< _Result, _Signature > >

Class template _Bind_result is always a bind expression.
The documentation for this struct was generated from the following file:

- functional

5.543 std::is_bind_expression< const volatile _Bind< _Signature > > Struct Template Reference

`#include <functional>`

5.543.1 Detailed Description

template<typename _Signature>
struct std::is_bind_expression< const volatile _Bind< _Signature > >

Class template _Bind is always a bind expression.
The documentation for this struct was generated from the following file:

- functional

5.544 std::is_bind_expression< const volatile _Bind_result< _Result, _Signature > > Struct Template Reference

`#include <functional>`

5.544.1 Detailed Description

template<typename _Result, typename _Signature>
struct std::is_bind_expression< const volatile _Bind_result< _Result, _Signature > >

Class template _Bind_result is always a bind expression.
The documentation for this struct was generated from the following file:

- functional

5.545 std::is_bind_expression< volatile _Bind< _Signature > > Struct Template Reference

`#include <functional>`

5.545.1 Detailed Description

template<typename _Signature>
struct std::is_bind_expression< volatile _Bind< _Signature > >

Class template _Bind is always a bind expression.
The documentation for this struct was generated from the following file:

- functional

5.546 std::is_bind_expression< volatile _Bind_result< _Result, _Signature > > Struct Template Reference

`#include <functional>`

5.546.1 Detailed Description

template<typename _Result, typename _Signature>
struct std::is_bind_expression< volatile _Bind_result< _Result, _Signature > >

Class template _Bind_result is always a bind expression.
The documentation for this struct was generated from the following file:

- functional

5.547 std::is_class< _Tp > Struct Template Reference

`#include <type_traits>`

5.547.1 Detailed Description

template<typename _Tp>
struct std::is_class< _Tp >

is_class
The documentation for this struct was generated from the following file:

- type_traits

5.548 std::is_compound< _Tp > Struct Template Reference

`#include <type_traits>`

5.548.1 Detailed Description

template<typename _Tp>
struct std::is_compound< _Tp >

is_compound
The documentation for this struct was generated from the following file:

- type_traits

5.549 std::is_const< _Tp > Struct Template Reference

`#include <type_traits>`

5.549.1 Detailed Description

template<typename _Tp>
struct std::is_const< _Tp >

is_const
The documentation for this struct was generated from the following file:

- type_traits

5.550 std::is_constructible< _Tp, _Args > Struct Template Reference

`#include <type_traits>`

5.550.1 Detailed Description

template<typename _Tp, typename... _Args>
struct std::is_constructible< _Tp, _Args >

is_constructible
The documentation for this struct was generated from the following file:

- type_traits

5.551 std::is_copy_assignable< _Tp > Struct Template Reference

`#include <type_traits>`

5.551.1 Detailed Description

template<typename _Tp>
struct std::is_copy_assignable< _Tp >

is_copy_assignable
The documentation for this struct was generated from the following file:

- type_traits

5.552 std::is_copy_constructible< _Tp > Struct Template Reference

`#include <type_traits>`

5.552.1 Detailed Description

template<typename _Tp>
struct std::is_copy_constructible< _Tp >

is_copy_constructible
The documentation for this struct was generated from the following file:

- type_traits

5.553 std::is_default_constructible< _Tp > Struct Template Reference

`#include <type_traits>`

5.553.1 Detailed Description

template<typename _Tp>
struct std::is_default_constructible< _Tp >

is_default_constructible
The documentation for this struct was generated from the following file:

- type_traits

5.554 std::is_destructible< _Tp > Struct Template Reference

`#include <type_traits>`

5.554.1 Detailed Description

template<typename _Tp>
struct std::is_destructible< _Tp >

is_destructible
The documentation for this struct was generated from the following file:

- type_traits

5.555 std::is_empty< _Tp > Struct Template Reference

`#include <type_traits>`

5.555.1 Detailed Description

template<**typename _Tp**>
struct std::is_empty< **_Tp** >

is_empty
The documentation for this struct was generated from the following file:

- type_traits

5.556 std::is_enum< _Tp > Struct Template Reference

`#include <type_traits>`

5.556.1 Detailed Description

template<**typename _Tp**>
struct std::is_enum< **_Tp** >

is_enum
The documentation for this struct was generated from the following file:

- type_traits

5.557 std::is_error_code_enum< _Tp > Struct Template Reference

`#include <system_error>`

5.557.1 Detailed Description

template<**typename _Tp**>
struct std::is_error_code_enum< **_Tp** >

is_error_code_enum
The documentation for this struct was generated from the following file:

- system_error

5.558 std::is_error_code_enum< future_errc > Struct Reference

`#include <future>`

5.558.1 Detailed Description

Specialization that allows `future_errc` to convert to `error_code`.
The documentation for this struct was generated from the following file:

- future

5.559 std::is_error_condition_enum< _Tp > Struct Template Reference

`#include <system_error>`

5.559.1 Detailed Description

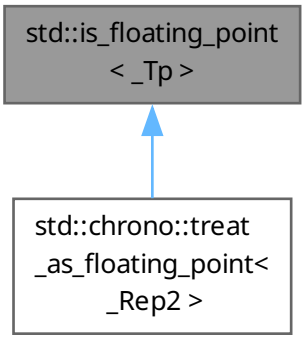
template<typename _Tp>
struct std::is_error_condition_enum< _Tp >

is_error_condition_enum
The documentation for this struct was generated from the following file:

- system_error

5.560 std::is_floating_point< _Tp > Struct Template Reference

`#include <type_traits>`
Inheritance diagram for std::is_floating_point< _Tp >:



5.560.1 Detailed Description

template<typename _Tp>
struct std::is_floating_point< _Tp >

is_floating_point
The documentation for this struct was generated from the following file:

- type_traits

5.561 std::is_function< _Tp > Struct Template Reference

`#include <type_traits>`

5.561.1 Detailed Description

template<typename _Tp>
struct std::is_function< _Tp >

is_function
The documentation for this struct was generated from the following file:

- type_traits

5.562 std::is_fundamental< _Tp > Struct Template Reference

`#include <type_traits>`

5.562.1 Detailed Description

template<typename _Tp>
struct std::is_fundamental< _Tp >

is_fundamental
The documentation for this struct was generated from the following file:

- type_traits

5.563 std::is_integral< _Tp > Struct Template Reference

`#include <type_traits>`

5.563.1 Detailed Description

template<typename _Tp>
struct std::is_integral< _Tp >

is_integral
The documentation for this struct was generated from the following file:

- type_traits

5.564 std::is_layout_compatible< _Tp, _Up > Struct Template Reference

`#include <type_traits>`

5.564.1 Detailed Description

template<typename _Tp, typename _Up>
struct std::is_layout_compatible< _Tp, _Up >

- Remove references and cv-qualifiers.

  Since

  C++20
  C++20

The documentation for this struct was generated from the following file:

- type_traits

5.565 std::is_literal_type< _Tp > Struct Template Reference

`#include <type_traits>`

5.565.1 Detailed Description

template<typename _Tp>
struct std::is_literal_type< _Tp >

is_literal_type

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

The documentation for this struct was generated from the following file:

- type_traits

5.566 std::is_lvalue_reference< typename > Struct Template Reference

`#include <type_traits>`

5.566.1 Detailed Description

template<typename>
struct std::is_lvalue_reference< typename >

is_lvalue_reference
The documentation for this struct was generated from the following file:

- type_traits

5.567 std::is_member_function_pointer< _Tp > Struct Template Reference

`#include <type_traits>`

5.567.1 Detailed Description

template<typename _Tp>
struct std::is_member_function_pointer< _Tp >

is_member_function_pointer
The documentation for this struct was generated from the following file:

- type_traits

5.568 std::is_member_object_pointer< _Tp > Struct Template Reference

`#include <type_traits>`

5.568.1 Detailed Description

template<typename _Tp>
struct std::is_member_object_pointer< _Tp >

is_member_object_pointer
The documentation for this struct was generated from the following file:

- type_traits

5.569 std::is_member_pointer< _Tp > Struct Template Reference

`#include <type_traits>`

5.569.1 Detailed Description

template<typename _Tp>
struct std::is_member_pointer< _Tp >

is_member_pointer
The documentation for this struct was generated from the following file:

- type_traits

5.570 std::is_move_assignable< _Tp > Struct Template Reference

`#include <type_traits>`

5.570.1 Detailed Description

template<typename _Tp>
struct std::is_move_assignable< _Tp >

is_move_assignable
The documentation for this struct was generated from the following file:

- type_traits

5.571 std::is_move_constructible< _Tp > Struct Template Reference

`#include <type_traits>`

5.571.1 Detailed Description

template<typename _Tp>
struct std::is_move_constructible< _Tp >

is_move_constructible
The documentation for this struct was generated from the following file:

- type_traits

5.572 std::is_nothrow_assignable< _Tp, _Up > Struct Template Reference

`#include <type_traits>`

5.572.1 Detailed Description

template<typename _Tp, typename _Up>
struct std::is_nothrow_assignable< _Tp, _Up >

is_nothrow_assignable
The documentation for this struct was generated from the following file:

- type_traits

5.573 std::is_nothrow_constructible< _Tp, _Args > Struct Template Reference

`#include <type_traits>`

5.573.1 Detailed Description

template<typename _Tp, typename... _Args>
struct std::is_nothrow_constructible< _Tp, _Args >

is_nothrow_constructible
The documentation for this struct was generated from the following file:

- type_traits

5.574 std::is_nothrow_copy_assignable< _Tp > Struct Template Reference

`#include <type_traits>`

5.574.1 Detailed Description

template<typename _Tp>
struct std::is_nothrow_copy_assignable< _Tp >

is_nothrow_copy_assignable
The documentation for this struct was generated from the following file:

- type_traits

5.575 std::is_nothrow_copy_constructible< _Tp > Struct Template Reference

`#include <type_traits>`

5.575.1 Detailed Description

template<typename _Tp>
struct std::is_nothrow_copy_constructible< _Tp >

is_nothrow_copy_constructible
The documentation for this struct was generated from the following file:

- type_traits

5.576 std::is_nothrow_default_constructible< _Tp > Struct Template Reference

`#include <type_traits>`

5.576.1 Detailed Description

template<typename _Tp>
struct std::is_nothrow_default_constructible< _Tp >

is_nothrow_default_constructible
The documentation for this struct was generated from the following file:

- type_traits

5.577 std::is_nothrow_destructible< _Tp > Struct Template Reference

`#include <type_traits>`

5.577.1 Detailed Description

template<typename _Tp>
struct std::is_nothrow_destructible< _Tp >

is_nothrow_destructible
The documentation for this struct was generated from the following file:

- type_traits

5.578 std::is_nothrow_move_assignable< _Tp > Struct Template Reference

`#include <type_traits>`

5.578.1 Detailed Description

template<**typename _Tp**>
struct std::is_nothrow_move_assignable< **_Tp** >

is_nothrow_move_assignable
The documentation for this struct was generated from the following file:

- type_traits

5.579 std::is_nothrow_move_constructible< _Tp > Struct Template Reference

`#include <type_traits>`

5.579.1 Detailed Description

template<**typename _Tp**>
struct std::is_nothrow_move_constructible< **_Tp** >

is_nothrow_move_constructible
The documentation for this struct was generated from the following file:

- type_traits

5.580 std::is_object< _Tp > Struct Template Reference

`#include <type_traits>`

5.580.1 Detailed Description

template<**typename _Tp**>
struct std::is_object< **_Tp** >

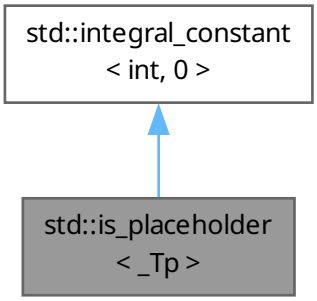
is_object
The documentation for this struct was generated from the following file:

- type_traits

5.581 std::is_placeholder< _Tp > Struct Template Reference

`#include <functional>`

Inheritance diagram for std::is_placeholder< _Tp >:



Public Types

- using **type**
- using **value_type**

Public Member Functions

- constexpr **operator value_type** () const noexcept

Static Public Attributes

- static constexpr int **value**

5.581.1 Detailed Description

template<typename _Tp>
struct std::is_placeholder< _Tp >

Determines if the given type _Tp is a placeholder in a bind() expression and, if so, which placeholder it is. C++11 [func.bind.isplace].

Since

C++11

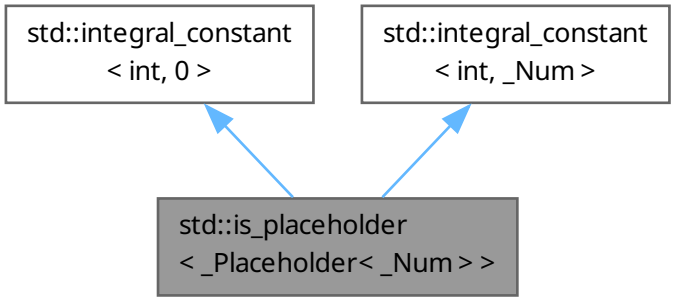
The documentation for this struct was generated from the following file:

- functional

5.582 std::is_placeholder< _Placeholder< _Num > > Struct Template Reference

`#include <functional>`

Inheritance diagram for std::is_placeholder< _Placeholder< _Num > >:



Public Types

- using **type**
- using **value_type**

Public Member Functions

- constexpr **operator value_type** () const noexcept

Static Public Attributes

- static constexpr int **value**

5.582.1 Detailed Description

template<**int _Num**>
struct std::is_placeholder< **_Placeholder**< **_Num** > >

Partial specialization of is_placeholder that provides the placeholder number for the placeholder objects defined by libstdc++.

Since

C++11

The documentation for this struct was generated from the following file:

- functional

5.583 std::is_pod< _Tp > Struct Template Reference

`#include <type_traits>`

5.583.1 Detailed Description

template<**typename _Tp**>
struct std::is_pod< **_Tp** >

is_pod

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

The documentation for this struct was generated from the following file:

- type_traits

5.584 std::is_pointer< _Tp > Struct Template Reference

`#include <type_traits>`

5.584.1 Detailed Description

template<typename _Tp>
struct std::is_pointer< _Tp >

is_pointer
The documentation for this struct was generated from the following file:

- type_traits

5.585 std::is_pointer_interconvertible_base_of< _Base, _Derived > Struct Template Reference

`#include <type_traits>`

5.585.1 Detailed Description

template<typename _Base, typename _Derived>
struct std::is_pointer_interconvertible_base_of< _Base, _Derived >

True if `_Derived` is standard-layout and has a base class of type `_Base`

Since

C++20

The documentation for this struct was generated from the following file:

- type_traits

5.586 std::is_polymorphic< _Tp > Struct Template Reference

`#include <type_traits>`

5.586.1 Detailed Description

template<typename _Tp>
struct std::is_polymorphic< _Tp >

is_polymorphic
The documentation for this struct was generated from the following file:

- type_traits

5.587 std::is_reference< _Tp > Struct Template Reference

`#include <type_traits>`

5.587.1 Detailed Description

template<**typename _Tp**>
struct std::is_reference< **_Tp** >

is_reference
The documentation for this struct was generated from the following file:

- type_traits

5.588 std::is_rvalue_reference< typename > Struct Template Reference

`#include <type_traits>`

5.588.1 Detailed Description

template<**typename**>
struct std::is_rvalue_reference< **typename** >

is_rvalue_reference
The documentation for this struct was generated from the following file:

- type_traits

5.589 std::is_same< _Tp, _Up > Struct Template Reference

`#include <type_traits>`

5.589.1 Detailed Description

template<**typename _Tp, typename _Up**>
struct std::is_same< **_Tp, _Up** >

is_same
The documentation for this struct was generated from the following file:

- type_traits

5.590 std::is_scalar< _Tp > Struct Template Reference

`#include <type_traits>`

5.590.1 Detailed Description

template<**typename _Tp**>
struct std::is_scalar< **_Tp** >

is_scalar
The documentation for this struct was generated from the following file:

- type_traits

5.591 std::is_signed< _Tp > Struct Template Reference

`#include <type_traits>`

5.591.1 Detailed Description

template<typename _Tp>
struct std::is_signed< _Tp >

is_signed
The documentation for this struct was generated from the following file:

- type_traits

5.592 std::is_standard_layout< _Tp > Struct Template Reference

`#include <type_traits>`

5.592.1 Detailed Description

template<typename _Tp>
struct std::is_standard_layout< _Tp >

is_standard_layout
The documentation for this struct was generated from the following file:

- type_traits

5.593 std::is_trivial< _Tp > Struct Template Reference

`#include <type_traits>`

5.593.1 Detailed Description

template<typename _Tp>
struct std::is_trivial< _Tp >

is_trivial

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

The documentation for this struct was generated from the following file:

- type_traits

5.594 std::is_trivially_assignable< _Tp, _Up > Struct Template Reference

`#include <type_traits>`

5.594.1 Detailed Description

template<typename _Tp, typename _Up>
struct std::is_trivially_assignable< _Tp, _Up >

is_trivially_assignable
The documentation for this struct was generated from the following file:

- type_traits

5.595 std::is_trivially_constructible< _Tp, _Args > Struct Template Reference

`#include <type_traits>`

5.595.1 Detailed Description

template<typename _Tp, typename... _Args>
struct std::is_trivially_constructible< _Tp, _Args >

is_trivially_constructible
The documentation for this struct was generated from the following file:

- type_traits

5.596 std::is_trivially_copy_assignable< _Tp > Struct Template Reference

`#include <type_traits>`

5.596.1 Detailed Description

template<typename _Tp>
struct std::is_trivially_copy_assignable< _Tp >

is_trivially_copy_assignable
The documentation for this struct was generated from the following file:

- type_traits

5.597 std::is_trivially_copy_constructible< _Tp > Struct Template Reference

`#include <type_traits>`

5.597.1 Detailed Description

template<typename _Tp>
struct std::is_trivially_copy_constructible< _Tp >

is_trivially_copy_constructible
The documentation for this struct was generated from the following file:

- type_traits

5.598 std::is_trivially_copyable< _Tp > Struct Template Reference

`#include <type_traits>`

5.598.1 Detailed Description

template<typename _Tp>
struct std::is_trivially_copyable< _Tp >

is_trivially_copyable
The documentation for this struct was generated from the following file:

- type_traits

5.599 std::is_trivially_default_constructible< _Tp > Struct Template Reference

`#include <type_traits>`

5.599.1 Detailed Description

template<typename _Tp>
struct std::is_trivially_default_constructible< _Tp >

is_trivially_default_constructible
The documentation for this struct was generated from the following file:

- type_traits

5.600 std::is_trivially_destructible< _Tp > Struct Template Reference

`#include <type_traits>`

5.600.1 Detailed Description

template<typename _Tp>
struct std::is_trivially_destructible< _Tp >

is_trivially_destructible
The documentation for this struct was generated from the following file:

- type_traits

5.601 std::is_trivially_move_assignable< _Tp > Struct Template Reference

`#include <type_traits>`

5.601.1 Detailed Description

template<typename _Tp>
struct std::is_trivially_move_assignable< _Tp >

is_trivially_move_assignable
The documentation for this struct was generated from the following file:

- type_traits

5.602 std::is_trivially_move_constructible< _Tp > Struct Template Reference

`#include <type_traits>`

5.602.1 Detailed Description

template<typename _Tp>
struct std::is_trivially_move_constructible< _Tp >

is_trivially_move_constructible
The documentation for this struct was generated from the following file:

- type_traits

5.603 std::is_union< _Tp > Struct Template Reference

`#include <type_traits>`

5.603.1 Detailed Description

template<**typename _Tp**>
struct std::is_union< **_Tp** >

is_union
The documentation for this struct was generated from the following file:

- type_traits

5.604 std::is_unsigned< _Tp > Struct Template Reference

`#include <type_traits>`

5.604.1 Detailed Description

template<**typename _Tp**>
struct std::is_unsigned< **_Tp** >

is_unsigned
The documentation for this struct was generated from the following file:

- type_traits

5.605 std::is_void< _Tp > Struct Template Reference

`#include <type_traits>`

5.605.1 Detailed Description

template<**typename _Tp**>
struct std::is_void< **_Tp** >

is_void
The documentation for this struct was generated from the following file:

- type_traits

5.606 std::is_volatile< _Tp > Struct Template Reference

`#include <type_traits>`

5.606.1 Detailed Description

template<**typename _Tp**>
struct std::is_volatile< **_Tp** >

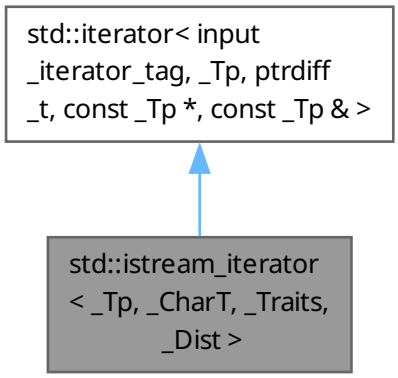
is_volatile
The documentation for this struct was generated from the following file:

- type_traits

5.607 std::istream_iterator< _Tp, _CharT, _Traits, _Dist > Class Template Reference

`#include <stream_iterator.h>`

Inheritance diagram for std::istream_iterator< _Tp, _CharT, _Traits, _Dist >:



Public Types

- typedef _CharT **char_type**
- typedef ptrdiff_t difference_type
- typedef basic_istream< _CharT, _Traits > **istream_type**
- typedef input_iterator_tag iterator_category
- typedef const _Tp * pointer
- typedef const _Tp & reference
- typedef _Traits **traits_type**
- typedef _Tp value_type

Public Member Functions

- constexpr istream_iterator () noexcept(/*conditional */)
- constexpr **istream_iterator** (const istream_iterator &__obj) noexcept(/*conditional */)
- istream_iterator (istream_type &__s)
- const _Tp & **operator*** () const noexcept
- istream_iterator & **operator++** ()
- istream_iterator **operator++** (int)
- const _Tp * **operator->** () const noexcept
- istream_iterator & **operator=** (const istream_iterator &)=default

Friends

- bool operator== (const istream_iterator &__x, const istream_iterator &__y) noexcept

5.607.1 Detailed Description

template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename _Dist = ptrdiff_t>
class std::istream_iterator< _Tp, _CharT, _Traits, _Dist >

Provides input iterator semantics for streams.

5.607.2 Member Typedef Documentation

difference_type

```
typedef ptrdiff_t std::iterator< input_iterator_tag, _Tp, ptrdiff_t, const _Tp *, const _Tp & >↩
::difference_type  [inherited]
```
Distance between iterators is represented as this type.

iterator_category

```
typedef input_iterator_tag std::iterator< input_iterator_tag, _Tp, ptrdiff_t, const _Tp *, const
_Tp & >::iterator_category  [inherited]
```
One of the tag types.

pointer

```
typedef const _Tp * std::iterator< input_iterator_tag, _Tp, ptrdiff_t, const _Tp *, const _Tp &
>::pointer  [inherited]
```
This type represents a pointer-to-value_type.

reference

```
typedef const _Tp & std::iterator< input_iterator_tag, _Tp, ptrdiff_t, const _Tp *, const _Tp &
>::reference  [inherited]
```
This type represents a reference-to-value_type.

value_type

```
typedef _Tp std::iterator< input_iterator_tag, _Tp, ptrdiff_t, const _Tp *, const _Tp & >::value↩
_type  [inherited]
```
The type "pointed to" by the iterator.

5.607.3 Constructor & Destructor Documentation

istream_iterator() [1/2]

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename
_Dist = ptrdiff_t>
std::istream_iterator< _Tp, _CharT, _Traits, _Dist >::istream_iterator ()  [inline], [constexpr],
[noexcept]
```
Construct end of input stream iterator.
Referenced by operator==.

istream_iterator() [2/2]

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename
_Dist = ptrdiff_t>
std::istream_iterator< _Tp, _CharT, _Traits, _Dist >::istream_iterator (
            istream_type & __s)  [inline]
```
Construct start of input stream iterator.
References std::__addressof().

5.607.4 Friends And Related Symbol Documentation

operator==

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename
```

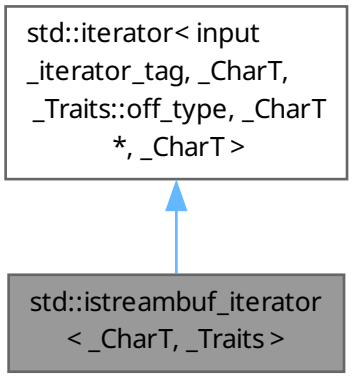
```
_Dist = ptrdiff_t>
bool operator== (
            const istream_iterator< _Tp, _CharT, _Traits, _Dist > & __x,
            const istream_iterator< _Tp, _CharT, _Traits, _Dist > & __y)  [friend]
```

Return true if the iterators refer to the same stream, or are both at end-of-stream.
References istream_iterator().
The documentation for this class was generated from the following file:

- stream_iterator.h

5.608 std::istreambuf_iterator< _CharT, _Traits > Class Template Reference

`#include <streambuf_iterator.h>`
Inheritance diagram for std::istreambuf_iterator< _CharT, _Traits >:



Public Types

- typedef _Traits::off_type difference_type
- typedef input_iterator_tag iterator_category
- typedef _CharT reference
- typedef _CharT value_type

- using pointer
- typedef _CharT char_type
- typedef _Traits traits_type
- typedef _Traits::int_type int_type
- typedef basic_streambuf< _CharT, _Traits > streambuf_type
- typedef basic_istream< _CharT, _Traits > istream_type

Public Member Functions

- constexpr istreambuf_iterator () noexcept
- **istreambuf_iterator** (const istreambuf_iterator &) noexcept=default

- istreambuf_iterator (istream_type &__s) noexcept
- istreambuf_iterator (streambuf_type *__s) noexcept
- bool equal (const istreambuf_iterator &__b) const
- char_type operator* () const
- istreambuf_iterator & operator++ ()
- istreambuf_iterator operator++ (int)
- istreambuf_iterator & **operator=** (const istreambuf_iterator &) noexcept=default

Friends

- template<bool _IsMove, typename _CharT2>
  __gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, _CharT2 * >::__type **__copy_move_a2** (istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, _CharT2 *)
- template<typename _CharT2, typename _Size>
  __gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, _CharT2 * >::__type **__copy_n_a** (istreambuf_iterator< _CharT2 >, _Size, _CharT2 *, bool)
- template<typename _CharT2, typename _Distance>
  __gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, void >::__type **advance** (istreambuf_iterator< _CharT2 > &, _Distance)
- template<typename _CharT2>
  __gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, ostreambuf_iterator< _CharT2 > >::__type **copy** (istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, ostreambuf_iterator< _CharT2 >)
- template<typename _CharT2>
  __gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, istreambuf_iterator< _CharT2 > >::__type **find** (istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, const _CharT2 &)

5.608.1 Detailed Description

template<typename _CharT, typename _Traits>
class std::istreambuf_iterator< _CharT, _Traits >

Provides input iterator semantics for streambufs.

5.608.2 Member Typedef Documentation

char_type

```
template<typename _CharT, typename _Traits>
typedef _CharT std::istreambuf_iterator< _CharT, _Traits >::char_type
```

Public typedefs.

difference_type

```
typedef _Traits::off_type std::iterator< input_iterator_tag, _CharT, _Traits::off_type, _CharT *,
_CharT >::difference_type  [inherited]
```

Distance between iterators is represented as this type.

int_type

```
template<typename _CharT, typename _Traits>
typedef _Traits::int_type std::istreambuf_iterator< _CharT, _Traits >::int_type
```

Public typedefs.

istream_type

```
template<typename _CharT, typename _Traits>
typedef basic_istream<_CharT, _Traits> std::istreambuf_iterator< _CharT, _Traits >::istream_type
```
Public typedefs.

iterator_category

```
typedef input_iterator_tag std::iterator< input_iterator_tag, _CharT, _Traits::off_type, _CharT *,
_CharT >::iterator_category  [inherited]
```
One of the tag types.

pointer

```
template<typename _CharT, typename _Traits>
using std::istreambuf_iterator< _CharT, _Traits >::pointer
```
Public typedefs.

reference

```
typedef _CharT std::iterator< input_iterator_tag, _CharT, _Traits::off_type, _CharT *, _CharT >↩
::reference  [inherited]
```
This type represents a reference-to-value_type.

streambuf_type

```
template<typename _CharT, typename _Traits>
typedef basic_streambuf<_CharT, _Traits> std::istreambuf_iterator< _CharT, _Traits >::streambuf↩
_type
```
Public typedefs.

traits_type

```
template<typename _CharT, typename _Traits>
typedef _Traits std::istreambuf_iterator< _CharT, _Traits >::traits_type
```
Public typedefs.

value_type

```
typedef _CharT std::iterator< input_iterator_tag, _CharT, _Traits::off_type, _CharT *, _CharT >↩
::value_type  [inherited]
```
The type "pointed to" by the iterator.

5.608.3 Constructor & Destructor Documentation

istreambuf_iterator() [1/3]

```
template<typename _CharT, typename _Traits>
std::istreambuf_iterator< _CharT, _Traits >::istreambuf_iterator ()  [inline], [constexpr], [noexcept]
```
Construct end of input stream iterator.
Referenced by equal(), and operator++().

istreambuf_iterator() [2/3]

```
template<typename _CharT, typename _Traits>
std::istreambuf_iterator< _CharT, _Traits >::istreambuf_iterator (
            istream_type & __s)  [inline], [noexcept]
```

Construct start of input stream iterator.

istreambuf_iterator() [3/3]

```
template<typename _CharT, typename _Traits>
std::istreambuf_iterator< _CharT, _Traits >::istreambuf_iterator (
            streambuf_type * __s)  [inline], [noexcept]
```

Construct start of streambuf iterator.

5.608.4 Member Function Documentation

equal()

```
template<typename _CharT, typename _Traits>
bool std::istreambuf_iterator< _CharT, _Traits >::equal (
            const istreambuf_iterator< _CharT, _Traits > & __b) const  [inline], [nodiscard]
```

Return true both iterators are end or both are not end.
References istreambuf_iterator().

operator*()

```
template<typename _CharT, typename _Traits>
char_type std::istreambuf_iterator< _CharT, _Traits >::operator* () const  [inline], [nodiscard]
```

Return the current character pointed to by iterator. This returns streambuf.sgetc(). It cannot be assigned. NB: The result of operator*() on an end of stream is undefined.

operator++() [1/2]

```
template<typename _CharT, typename _Traits>
istreambuf_iterator & std::istreambuf_iterator< _CharT, _Traits >::operator++ ()  [inline]
```

Advance the iterator. Calls streambuf.sbumpc().

operator++() [2/2]

```
template<typename _CharT, typename _Traits>
istreambuf_iterator std::istreambuf_iterator< _CharT, _Traits >::operator++ (
            int )  [inline]
```

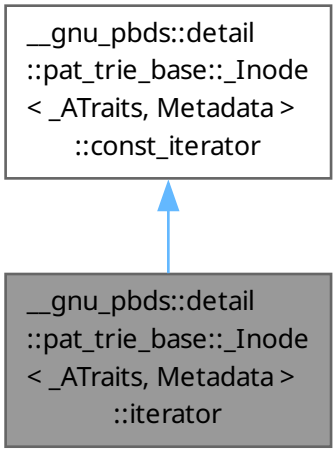
Advance the iterator. Calls streambuf.sbumpc().
References istreambuf_iterator().
The documentation for this class was generated from the following files:

- iosfwd
- streambuf_iterator.h

5.609 __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >::iterator Struct Reference

```
#include <pat_trie_base.hpp>
```

Inheritance diagram for __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >::iterator:



Public Types

- typedef _Alloc::difference_type **difference_type**
- typedef std::forward_iterator_tag **iterator_category**
- typedef node_pointer_pointer **pointer**
- typedef node_pointer_reference **reference**
- typedef node_pointer **value_type**

Public Member Functions

- **iterator** (node_pointer_pointer p_p_cur=0, node_pointer_pointer p_p_end=0)
- bool **operator!=** (const const_iterator &other) const
- bool **operator!=** (const iterator &other) const
- node_const_pointer **operator*** () const
- node_pointer **operator*** ()
- iterator & **operator++** ()
- iterator **operator++** (int)
- const node_pointer_pointer **operator->** () const
- node_pointer_pointer **operator->** ()
- bool **operator==** (const const_iterator &other) const
- bool **operator==** (const iterator &other) const

Public Attributes

- node_pointer_pointer **m_p_p_cur**
- node_pointer_pointer **m_p_p_end**

5.609.1 Detailed Description

template<typename _ATraits, typename Metadata>
struct __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >::iterator

Child iterator.
The documentation for this struct was generated from the following file:

- pat_trie_base.hpp

5.610 std::experimental::filesystem::v1::path::iterator Class Reference

`#include <fs_path.h>`

Public Types

- using **difference_type**
- using **iterator_category**
- using **pointer**
- using **reference**
- using **value_type**

Public Member Functions

- **iterator** (const iterator &)=default
- reference **operator*** () const noexcept
- iterator & **operator++** () noexcept
- iterator **operator++** (int) noexcept
- iterator & **operator--** () noexcept
- iterator **operator--** (int) noexcept
- pointer **operator->** () const noexcept
- iterator & **operator=** (const iterator &)=default

Friends

- bool **operator!=** (const iterator &__lhs, const iterator &__rhs) noexcept
- bool **operator==** (const iterator &__lhs, const iterator &__rhs) noexcept
- class **path**

5.610.1 Detailed Description

An iterator for the components of a path.
The documentation for this class was generated from the following file:

- experimental/bits/fs_path.h

5.611 std::filesystem::path::iterator Class Reference

`#include <filesystem>`

Public Types

- using **difference_type**
- using **iterator_category**
- using **pointer**
- using **reference**
- using **value_type**

Public Member Functions

- **iterator** (const iterator &)=default
- reference **operator*** () const noexcept
- iterator & **operator++** () noexcept
- iterator **operator++** (int) noexcept
- iterator & **operator--** () noexcept
- iterator **operator--** (int) noexcept
- pointer **operator->** () const noexcept
- iterator & **operator=** (const iterator &)=default

Friends

- void **__path_iter_advance** (iterator &__i, difference_type __n) noexcept
- difference_type **__path_iter_distance** (const iterator &__first, const iterator &__last) noexcept
- bool **operator!=** (const iterator &__lhs, const iterator &__rhs) noexcept
- bool **operator==** (const iterator &__lhs, const iterator &__rhs) noexcept
- class **path**

5.611.1 Detailed Description

An iterator for the components of a path.

Since

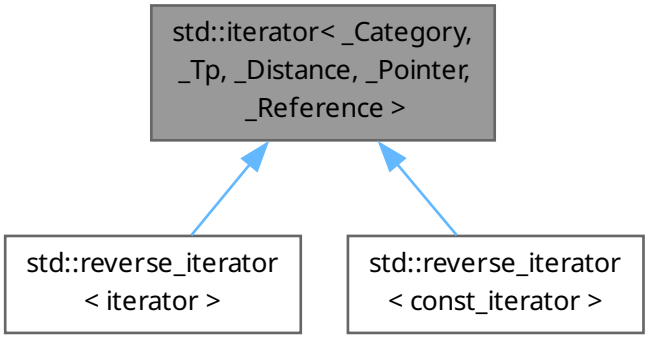
C++17

The documentation for this class was generated from the following file:

- bits/fs_path.h

5.612 std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference > Struct Template Reference

`#include <stl_iterator_base_types.h>`
Inheritance diagram for std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >:



Public Types

- typedef _Distance difference_type
- typedef _Category iterator_category
- typedef _Pointer pointer
- typedef _Reference reference
- typedef _Tp value_type

5.612.1 Detailed Description

template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*, typename _Reference = _Tp&>
struct std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >

Common iterator class.
This class does nothing but define nested typedefs. Iterator classes can inherit from this class to save some work. The typedefs are then used in specializations and overloading.
In particular, there are no default implementations of requirements such as `operator++` and the like. (How could there be?)

5.612.2 Member Typedef Documentation

difference_type

```
template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _↩
Tp*, typename _Reference = _Tp&>
typedef _Distance std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::difference_↩
type
```

Distance between iterators is represented as this type.

iterator_category

```
template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _↩
Tp*, typename _Reference = _Tp&>
typedef _Category std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::iterator_↩
category
```

One of the tag types.

pointer

```
template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _↩
Tp*, typename _Reference = _Tp&>
typedef _Pointer std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::pointer
```

This type represents a pointer-to-value_type.

reference

```
template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _↩
Tp*, typename _Reference = _Tp&>
typedef _Reference std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::reference
```

This type represents a reference-to-value_type.

value_type

```
template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _↩
Tp*, typename _Reference = _Tp&>
typedef _Tp std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::value_type
```

The type "pointed to" by the iterator.
The documentation for this struct was generated from the following file:

- stl_iterator_base_types.h

5.613 std::iterator_traits< _Iterator > Struct Template Reference

5.613.1 Detailed Description

template<typename _Iterator>
struct std::iterator_traits< _Iterator >

Traits class for iterators.
This class does nothing but define nested typedefs. The general version simply *forwards* the nested typedefs from the Iterator argument. Specialized versions for pointers and pointers-to-const provide tighter, more correct semantics.
The documentation for this struct was generated from the following file:

- stl_iterator_base_types.h

5.614 std::iterator_traits< _Tp * > Struct Template Reference

`#include <stl_iterator_base_types.h>`

Public Types

- using **difference_type**
- using **iterator_category**
- using **iterator_concept**
- using **pointer**
- using **reference**
- using **value_type**

5.614.1 Detailed Description

template<typename _Tp>
requires is_object_v<_Tp>
struct std::iterator_traits< _Tp * >

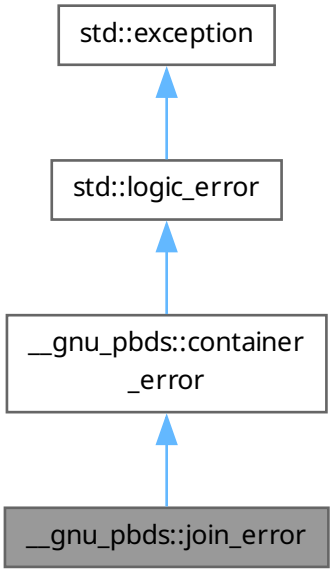
Partial specialization for object pointer types.
The documentation for this struct was generated from the following file:

- stl_iterator_base_types.h

5.615 __gnu_pbds::join_error Struct Reference

`#include <exception.hpp>`

Inheritance diagram for __gnu_pbds::join_error:



Public Member Functions

- virtual const char ∗ what () const noexcept

5.615.1 Detailed Description

A join cannot be performed logical reasons (i.e., the ranges of the two container objects being joined overlaps.

5.615.2 Member Function Documentation

what()

```
virtual const char * std::logic_error::what () const  [virtual], [noexcept], [inherited]
```

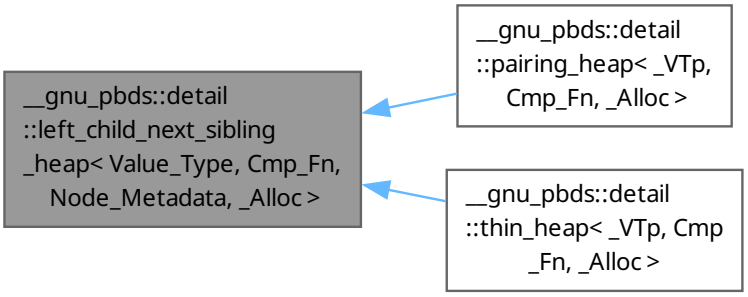
Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).
Reimplemented from std::exception.
Reimplemented in std::future_error.
The documentation for this struct was generated from the following file:

- exception.hpp

5.616 __gnu_pbds::detail::left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc > Class Template Reference

```
#include <left_child_next_sibling_heap_.hpp>
```

Inheritance diagram for __gnu_pbds::detail::left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _↩ Alloc >:



Public Types

- typedef _Alloc **allocator_type**
- typedef Cmp_Fn **cmp_fn**
- typedef left_child_next_sibling_heap_const_iterator_< node, _Alloc > **const_iterator**
- typedef __rebind_v::const_pointer **const_pointer**
- typedef __rebind_v::const_reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef const_iterator **iterator**
- typedef left_child_next_sibling_heap_node_< Value_Type, Node_Metadata, _Alloc > **node**
- typedef left_child_next_sibling_heap_node_point_const_iterator_< node, _Alloc > **point_const_iterator**
- typedef point_const_iterator **point_iterator**
- typedef __rebind_v::pointer **pointer**
- typedef __rebind_v::reference **reference**
- typedef _Alloc::size_type **size_type**
- typedef Value_Type **value_type**

Public Member Functions

- **left_child_next_sibling_heap** (const Cmp_Fn &)
- **left_child_next_sibling_heap** (const left_child_next_sibling_heap &)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- Cmp_Fn & **get_cmp_fn** ()
- const Cmp_Fn & **get_cmp_fn** () const
- size_type **max_size** () const
- size_type **size** () const
- void **swap** (left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc > &)

Protected Types

- typedef alloc_traits::allocator_type **node_allocator**
- typedef alloc_traits::const_pointer **node_const_pointer**
- typedef Node_Metadata **node_metadata**
- typedef alloc_traits::pointer **node_pointer**
- typedef std::pair< node_pointer, node_pointer > **node_pointer_pair**

Protected Member Functions

- void **actual_erase_node** (node_pointer)
- void **bubble_to_top** (node_pointer)
- void **clear_imp** (node_pointer)
- node_pointer **get_new_node_for_insert** (const_reference)
- template<typename Pred>
  node_pointer **prune** (Pred)
- void **swap_with_parent** (node_pointer, node_pointer)
- void **to_linked_list** ()
- void **value_swap** (left_child_next_sibling_heap &)

Static Protected Member Functions

- static void **make_child_of** (node_pointer, node_pointer)
- static node_pointer **parent** (node_pointer)

Protected Attributes

- node_pointer **m_p_root**
- size_type **m_size**

5.616.1 Detailed Description

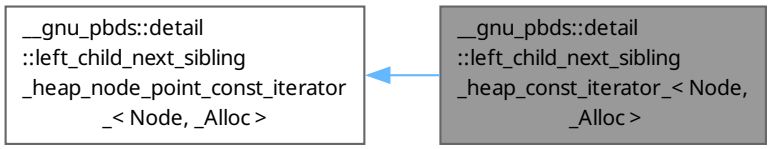
template<typename Value_Type, typename Cmp_Fn, typename Node_Metadata, typename _Alloc>
class __gnu_pbds::detail::left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >

Base class for a basic heap.
The documentation for this class was generated from the following file:

- left_child_next_sibling_heap_.hpp

5.617 __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc > Class Template Reference

`#include <const_iterator.hpp>`
Inheritance diagram for __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >:



Public Types

- typedef base_type::const_pointer const_pointer
- typedef base_type::const_reference const_reference
- typedef _Alloc::difference_type difference_type
- typedef std::forward_iterator_tag iterator_category
- typedef base_type::pointer pointer
- typedef base_type::reference reference
- typedef base_type::value_type value_type

Public Member Functions

- left_child_next_sibling_heap_const_iterator_ ()
- left_child_next_sibling_heap_const_iterator_ (const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > &other)
- **left_child_next_sibling_heap_const_iterator_** (node_pointer p_nd)
- bool operator!= (const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > &other) const
- bool operator!= (const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &other) const
- const_reference operator* () const
- left_child_next_sibling_heap_const_iterator_< Node, _Alloc > & **operator++** ()
- left_child_next_sibling_heap_const_iterator_< Node, _Alloc > **operator++** (int)
- const_pointer operator-> () const
- bool operator== (const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > &other) const
- bool operator== (const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &other) const

Public Attributes

- node_pointer **m_p_nd**

5.617.1 Detailed Description

template<typename Node, typename _Alloc>
class __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >

Const point-type iterator.

5.617.2 Member Typedef Documentation

const_pointer

```
template<typename Node, typename _Alloc>
typedef base_type::const_pointer __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_<
Node, _Alloc >::const_pointer
```

Iterator's const pointer type.

const_reference

```
template<typename Node, typename _Alloc>
typedef base_type::const_reference __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_<
Node, _Alloc >::const_reference
```

Iterator's const reference type.

difference_type

```
template<typename Node, typename _Alloc>
typedef _Alloc::difference_type __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_<
Node, _Alloc >::difference_type
```
Difference type.

iterator_category

```
template<typename Node, typename _Alloc>
typedef std::forward_iterator_tag __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_<
Node, _Alloc >::iterator_category
```
Category.

pointer

```
template<typename Node, typename _Alloc>
typedef base_type::pointer __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_<
Node, _Alloc >::pointer
```
Iterator's pointer type.

reference

```
template<typename Node, typename _Alloc>
typedef base_type::reference __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_<
Node, _Alloc >::reference
```
Iterator's reference type.

value_type

```
template<typename Node, typename _Alloc>
typedef base_type::value_type __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_<
Node, _Alloc >::value_type
```
Iterator's value type.

5.617.3 Constructor & Destructor Documentation

left_child_next_sibling_heap_const_iterator_() [1/2]

```
template<typename Node, typename _Alloc>
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::left_child_↩
next_sibling_heap_const_iterator_ ()  [inline]
```
Default constructor.

left_child_next_sibling_heap_const_iterator_() [2/2]

```
template<typename Node, typename _Alloc>
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::left_child_↩
next_sibling_heap_const_iterator_ (
            const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > & other)  [inline]
```
Copy constructor.

5.617.4 Member Function Documentation

operator"!=() [1/2]

```
template<typename Node, typename _Alloc>
```

```
bool __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::operator!=
(
            const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > & other) const
[inline]
```

Compares content (negatively) to a different iterator object.

operator"!=() [2/2]

```
template<typename Node, typename _Alloc>
bool __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc
>::operator!= (
            const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &
other) const  [inline], [inherited]
```

Compares content (negatively) to a different iterator object.

operator*()

```
template<typename Node, typename _Alloc>
const_reference __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_<
Node, _Alloc >::operator* () const  [inline], [inherited]
```

Access.

operator->()

```
template<typename Node, typename _Alloc>
const_pointer __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node,
_Alloc >::operator-> () const  [inline], [inherited]
```

Access.

operator==() [1/2]

```
template<typename Node, typename _Alloc>
bool __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::operator==
(
            const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > & other) const
[inline]
```

Compares content to a different iterator object.

operator==() [2/2]

```
template<typename Node, typename _Alloc>
bool __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc
>::operator== (
            const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &
other) const  [inline], [inherited]
```

Compares content to a different iterator object.

The documentation for this class was generated from the following file:

- left_child_next_sibling_heap_/const_iterator.hpp

5.618 __gnu_pbds::detail::left_child_next_sibling_heap_node_< _Value, _Metadata, _Alloc > Struct Template Reference

```
#include <node.hpp>
```

Public Types

- typedef _Metadata **metadata_type**
- typedef rebind_traits< _Alloc, this_type >::pointer **node_pointer**
- typedef _Alloc::size_type **size_type**
- typedef _Value **value_type**

Public Attributes

- metadata_type **m_metadata**
- node_pointer **m_p_l_child**
- node_pointer **m_p_next_sibling**
- node_pointer **m_p_prev_or_parent**
- value_type **m_value**

5.618.1 Detailed Description

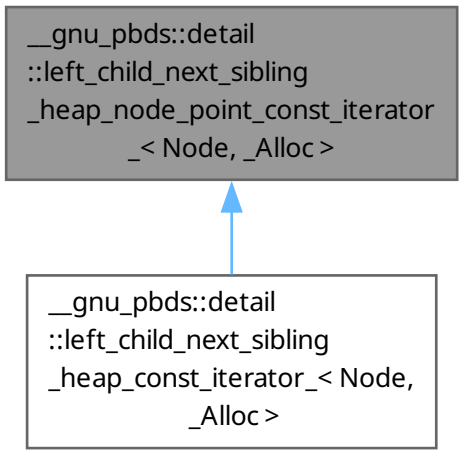
template<typename _Value, typename _Metadata, typename _Alloc>
struct __gnu_pbds::detail::left_child_next_sibling_heap_node_< _Value, _Metadata, _Alloc >

Node.
The documentation for this struct was generated from the following file:

- left_child_next_sibling_heap_/node.hpp

5.619 __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > Class Template Reference

`#include <point_const_iterator.hpp>`
Inheritance diagram for __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >:



Public Types

- typedef rebind_traits< _Alloc, value_type >::const_pointer const_pointer
- typedef rebind_traits< _Alloc, value_type >::const_reference const_reference
- typedef trivial_iterator_difference_type difference_type
- typedef trivial_iterator_tag iterator_category
- typedef rebind_traits< _Alloc, value_type >::pointer pointer
- typedef rebind_traits< _Alloc, value_type >::reference reference
- typedef Node::value_type value_type

Public Member Functions

- left_child_next_sibling_heap_node_point_const_iterator_ ()
- left_child_next_sibling_heap_node_point_const_iterator_ (const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &other)
- **left_child_next_sibling_heap_node_point_const_iterator_** (node_pointer p_nd)
- bool operator!= (const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &other) const
- const_reference operator* () const
- const_pointer operator-> () const
- bool operator== (const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &other) const

Public Attributes

- node_pointer **m_p_nd**

Protected Types

- typedef rebind_traits< _Alloc, Node >::pointer **node_pointer**

5.619.1 Detailed Description

template<typename Node, typename _Alloc>
class __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >

Const point-type iterator.

5.619.2 Member Typedef Documentation

const_pointer

```
template<typename Node, typename _Alloc>
typedef rebind_traits<_Alloc,value_type>::const_pointer __gnu_pbds::detail::left_child_next_sibling_heap_node_poi
Node, _Alloc >::const_pointer
```

Iterator's const pointer type.

const_reference

```
template<typename Node, typename _Alloc>
typedef rebind_traits<_Alloc,value_type>::const_reference __gnu_pbds::detail::left_child_next_sibling_heap_node_p
Node, _Alloc >::const_reference
```

Iterator's const reference type.

difference_type

```
template<typename Node, typename _Alloc>
typedef trivial_iterator_difference_type __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterat
Node, _Alloc >::difference_type
```
Difference type.

iterator_category

```
template<typename Node, typename _Alloc>
typedef trivial_iterator_tag __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_<
Node, _Alloc >::iterator_category
```
Category.

pointer

```
template<typename Node, typename _Alloc>
typedef rebind_traits<_Alloc,value_type>::pointer __gnu_pbds::detail::left_child_next_sibling_heap_node_point_co
Node, _Alloc >::pointer
```
Iterator's pointer type.

reference

```
template<typename Node, typename _Alloc>
typedef rebind_traits<_Alloc,value_type>::reference __gnu_pbds::detail::left_child_next_sibling_heap_node_point_
Node, _Alloc >::reference
```
Iterator's reference type.

value_type

```
template<typename Node, typename _Alloc>
typedef Node::value_type __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_<
Node, _Alloc >::value_type
```
Iterator's value type.

5.619.3 Constructor & Destructor Documentation

left_child_next_sibling_heap_node_point_const_iterator_() [1/2]

```
template<typename Node, typename _Alloc>
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >↩
::left_child_next_sibling_heap_node_point_const_iterator_ ()  [inline]
```
Default constructor.

left_child_next_sibling_heap_node_point_const_iterator_() [2/2]

```
template<typename Node, typename _Alloc>
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >↩
::left_child_next_sibling_heap_node_point_const_iterator_ (
            const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &
other)  [inline]
```
Copy constructor.

5.619.4 Member Function Documentation

operator"!=()

```
template<typename Node, typename _Alloc>
bool __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc
>::operator!= (
            const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &
other) const  [inline]
```

Compares content (negatively) to a different iterator object.

operator*()

```
template<typename Node, typename _Alloc>
const_reference __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_<
Node, _Alloc >::operator* () const  [inline]
```

Access.

operator->()

```
template<typename Node, typename _Alloc>
const_pointer __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node,
_Alloc >::operator-> () const  [inline]
```

Access.

operator==()

```
template<typename Node, typename _Alloc>
bool __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc
>::operator== (
            const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &
other) const  [inline]
```

Compares content to a different iterator object.

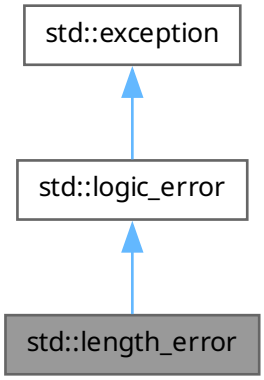
The documentation for this class was generated from the following file:

- left_child_next_sibling_heap_/point_const_iterator.hpp

5.620 std::length_error Class Reference

```
#include <stdexcept>
```

Inheritance diagram for std::length_error:



Public Member Functions

- **length_error** (const char *)
- **length_error** (const length_error &)=default
- **length_error** (const string &__arg)
- **length_error** (length_error &&)=default
- length_error & **operator=** (const length_error &)=default
- length_error & **operator=** (length_error &&)=default
- virtual const char * what () const noexcept

5.620.1 Detailed Description

Thrown when an object is constructed that would exceed its maximum permitted size (e.g., a basic_string instance).

5.620.2 Member Function Documentation

what()

```
virtual const char * std::logic_error::what () const  [virtual], [noexcept], [inherited]
```

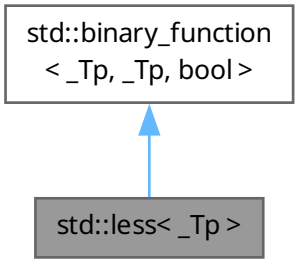
Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).
Reimplemented from std::exception.
Reimplemented in std::future_error.
The documentation for this class was generated from the following file:

- stdexcept

5.621 std::less< _Tp > Struct Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::less< _Tp >:



Public Types

- typedef _Tp first_argument_type
- typedef bool result_type
- typedef _Tp second_argument_type

Public Member Functions

- constexpr bool **operator()** (const _Tp &__x, const _Tp &__y) const

5.621.1 Detailed Description

template<typename _Tp>
struct std::less< _Tp >

One of the comparison functors.

5.621.2 Member Typedef Documentation

first_argument_type

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type  [inherited]
```

`first_argument_type` is the type of the first argument

result_type

```
typedef bool std::binary_function< _Tp, _Tp, bool >::result_type  [inherited]
```

`result_type` is the return type

second_argument_type

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type  [inherited]
```

`second_argument_type` is the type of the second argument

The documentation for this struct was generated from the following file:

- stl_function.h

5.622 std::ranges::less Struct Reference

`#include <ranges_cmp.h>`

Public Types

- using **is_transparent**

Public Member Functions

- template<typename _Tp, typename _Up>
  requires totally_ordered_with<_Tp, _Up>
  constexpr bool **operator()** (_Tp &&__t, _Up &&__u) const noexcept(noexcept(std::declval< _Tp >()< std::declval< _Up >()))

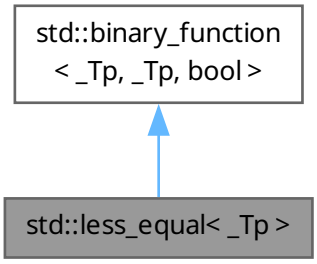
5.622.1 Detailed Description

ranges::less function object type.
The documentation for this struct was generated from the following file:

- ranges_cmp.h

5.623 std::less_equal< _Tp > Struct Template Reference

`#include <stl_function.h>`
Inheritance diagram for std::less_equal< _Tp >:



Public Types

- typedef _Tp first_argument_type
- typedef bool result_type
- typedef _Tp second_argument_type

Public Member Functions

- constexpr bool **operator()** (const _Tp &__x, const _Tp &__y) const

5.623.1 Detailed Description

template<typename _Tp>
struct std::less_equal< _Tp >

One of the comparison functors.

5.623.2 Member Typedef Documentation

first_argument_type

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type  [inherited]
```

`first_argument_type` is the type of the first argument

result_type

```
typedef bool std::binary_function< _Tp, _Tp, bool >::result_type  [inherited]
```

`result_type` is the return type

second_argument_type

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type  [inherited]
```

`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

5.624 std::ranges::less_equal Struct Reference

```
#include <ranges_cmp.h>
```

Public Types

- using **is_transparent**

Public Member Functions

- template<typename _Tp, typename _Up>
  requires totally_ordered_with<_Tp, _Up>
  constexpr bool **operator()** (_Tp &&__t, _Up &&__u) const noexcept(noexcept(std::declval< _Up >()< std::declval< _Tp >()))

5.624.1 Detailed Description

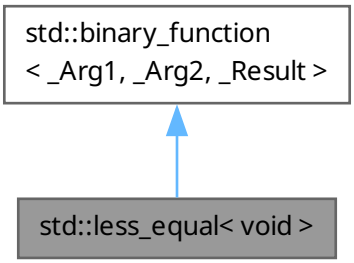
ranges::less_equal function object type.
The documentation for this struct was generated from the following file:

- ranges_cmp.h

5.625 std::less_equal< void > Struct Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::less_equal< void >:



Public Types

- typedef void first_argument_type
- typedef __is_transparent **is_transparent**
- typedef bool result_type
- typedef void second_argument_type

Public Member Functions

- template<typename _Tp, typename _Up>
  constexpr auto **operator()** (_Tp &&__t, _Up &&__u) const noexcept(noexcept(std::forward< _Tp >(__t)<=std::forward< _Up >(__u))) -> decltype(std::forward< _Tp >(__t)<=std::forward< _Up >(__u))
- template<typename _Tp, typename _Up>
  constexpr bool **operator()** (_Tp *__t, _Up *__u) const noexcept
- constexpr bool **operator()** (const void &__x, const void &__y) const

5.625.1 Detailed Description

One of the comparison functors.

5.625.2 Member Typedef Documentation

first_argument_type

```
typedef void std::binary_function< void, void, bool >::first_argument_type
```

`first_argument_type` is the type of the first argument

result_type

```
typedef bool std::binary_function< void, void, bool >::result_type
```

`result_type` is the return type

second_argument_type

```
typedef void std::binary_function< void, void, bool >::second_argument_type
```

`second_argument_type` is the type of the second argument

The documentation for this struct was generated from the following file:

- stl_function.h

5.626 __gnu_cxx::limit_condition::limit_adjustor Struct Reference

`#include <throw_allocator.h>`

Public Member Functions

- **limit_adjustor** (const size_t __l)

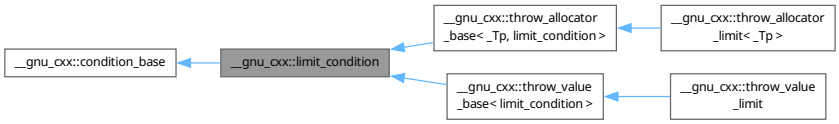
5.626.1 Detailed Description

Enter the nth condition.
The documentation for this struct was generated from the following file:

- throw_allocator.h

5.627 __gnu_cxx::limit_condition Struct Reference

`#include <throw_allocator.h>`
Inheritance diagram for __gnu_cxx::limit_condition:



Classes

- struct always_adjustor
- struct limit_adjustor
- struct never_adjustor

Static Public Member Functions

- static size_t & **count** ()
- static size_t & **limit** ()
- static void **set_limit** (const size_t __l)
- static void **throw_conditionally** ()

5.627.1 Detailed Description

Base class for incremental control and throw.
The documentation for this struct was generated from the following file:

- throw_allocator.h

5.628 std::linear_congruential_engine< _UIntType, __a, __c, __m > Class Template Reference

`#include <random>`

Public Types

- typedef _UIntType result_type

Public Member Functions

- linear_congruential_engine ()
- template<typename _Sseq, typename = _If_seed_seq<_Sseq>>
  linear_congruential_engine (_Sseq &__q)
- linear_congruential_engine (result_type __s)
- void discard (unsigned long long __z)
- result_type operator() ()
- template<typename _Sseq>
  _If_seed_seq< _Sseq > seed (_Sseq &__q)
- template<typename _Sseq>
  auto seed (_Sseq &__q) -> _If_seed_seq< _Sseq >
- void seed (result_type __s=default_seed)

Static Public Member Functions

- static constexpr result_type max ()
- static constexpr result_type min ()

Static Public Attributes

- static constexpr result_type **default_seed**
- static constexpr result_type increment
- static constexpr result_type modulus
- static constexpr result_type multiplier

Friends

- template<typename _UIntType1, _UIntType1 __a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::linear_congruential_engine< _UIntType1, __a1, __c1, __m1 > &__lcr)
- bool operator== (const linear_congruential_engine &__lhs, const linear_congruential_engine &__rhs)
- template<typename _UIntType1, _UIntType1 __a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::linear_congruential_engine< _UIntType1, __a1, __c1, __m1 > &__lcr)

5.628.1 Detailed Description

template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
class std::linear_congruential_engine< _UIntType, __a, __c, __m >

A model of a linear congruential random number generator.
A random number generator that produces pseudorandom numbers via linear function:

$$x_{i+1} \leftarrow (ax_i + c) \bmod m$$

The template parameter `_UIntType` must be an unsigned integral type large enough to store values up to (__m-1). If the template parameter `__m` is 0, the modulus `__m` used is std::numeric_limits<_UIntType>::max() plus 1. Otherwise, the template parameters `__a` and `__c` must be less than `__m`.
The size of the state is 1.

Since

C++11

5.628.2 Member Typedef Documentation

result_type

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
typedef _UIntType std::linear_congruential_engine< _UIntType, __a, __c, __m >::result_type
```

The type of the generated random value.

5.628.3 Constructor & Destructor Documentation

linear_congruential_engine() [1/3]

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
std::linear_congruential_engine< _UIntType, __a, __c, __m >::linear_congruential_engine ()  [inline]
```

Constructs a linear_congruential_engine random number generator engine with seed 1.

linear_congruential_engine() [2/3]

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
std::linear_congruential_engine< _UIntType, __a, __c, __m >::linear_congruential_engine (
            result_type __s)  [inline], [explicit]
```

Constructs a linear_congruential_engine random number generator engine with seed __s. The default seed value is 1.

Parameters

_↩ _s	The initial seed value.

linear_congruential_engine() [3/3]

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
template<typename _Sseq, typename = _If_seed_seq<_Sseq>>
std::linear_congruential_engine< _UIntType, __a, __c, __m >::linear_congruential_engine (
            _Sseq & __q)  [inline], [explicit]
```

Constructs a linear_congruential_engine random number generator engine seeded from the seed sequence __q.

Parameters

_↩ _q	the seed sequence.

5.628.4 Member Function Documentation

discard()

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
void std::linear_congruential_engine< _UIntType, __a, __c, __m >::discard (
            unsigned long long __z)  [inline]
```

Discard a sequence of random numbers.

max()

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
static constexpr result_type std::linear_congruential_engine< _UIntType, __a, __c, __m >::max ()
[inline], [static], [constexpr]
```

Gets the largest possible value in the output range.

min()

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
static constexpr result_type std::linear_congruential_engine< _UIntType, __a, __c, __m >::min ()
[inline], [static], [constexpr]
```

Gets the smallest possible value in the output range.
The minimum depends on the __c parameter: if it is zero, the minimum generated must be > 0, otherwise 0 is allowed.

operator()()

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
result_type std::linear_congruential_engine< _UIntType, __a, __c, __m >::operator() ()  [inline]
```

Gets the next random number in the sequence.

seed() [1/3]

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
template<typename _Sseq>
_If_seed_seq< _Sseq > std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed (
            _Sseq & __q)
```

Reseeds the linear_congruential_engine random number generator engine sequence using values from the seed sequence __q.

Parameters

_↩ _q	the seed sequence.

seed() [2/3]

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
template<typename _Sseq>
auto std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed (
            _Sseq & __q) -> _If_seed_seq<_Sseq>
```

Seeds the LCR engine with a value generated by __q.

seed() [3/3]

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
void std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed (
            result_type __s = default_seed)
```

Reseeds the linear_congruential_engine random number generator engine sequence to the seed __s.

Parameters

_↩ _s	The new seed.

Seeds the LCR with integral value __s, adjusted so that the ring identity is never a member of the convergence set.
Referenced by std::linear_congruential_engine< uint_fast32_t, 16807UL, 0UL, 2147483647UL >::linear_congruential_engine(), and std::linear_congruential_engine< uint_fast32_t, 16807UL, 0UL, 2147483647UL >::linear_congruential_engine().

5.628.5 Friends And Related Symbol Documentation

operator<<

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
template<typename _UIntType1, _UIntType1 __a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT,
typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::linear_congruential_engine< _UIntType1, __a1, __c1, __m1 > & __lcr)  [friend]
```

Writes the textual representation of the state x(i) of x to __os.

Parameters

__os	The output stream.
__lcr	A % linear_congruential_engine random number generator.

Returns

__os.

operator==

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
bool operator== (
            const linear_congruential_engine< _UIntType, __a, __c, __m > & __lhs,
            const linear_congruential_engine< _UIntType, __a, __c, __m > & __rhs)  [friend]
```

Compares two linear congruential random number generator objects of the same type for equality.

Parameters

__lhs	A linear congruential random number generator object.
__rhs	Another linear congruential random number generator object.

Returns

true if the infinite sequences of generated values would be equal, false otherwise.

operator>>

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
template<typename _UIntType1, _UIntType1 __a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT,
typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            std::linear_congruential_engine< _UIntType1, __a1, __c1, __m1 > & __lcr)  [friend]
```

Sets the state of the engine by reading its textual representation from __is.
The textual representation must have been previously written using an output stream whose imbued locale and whose type's template specialization arguments _CharT and _Traits were the same as those of __is.

Parameters

__is	The input stream.
__lcr	A % linear_congruential_engine random number generator.

Returns

__is.

5.628.6 Member Data Documentation

increment

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
result_type std::linear_congruential_engine< _UIntType, __a, __c, __m >::increment  [static],
[constexpr]
```

An increment.

modulus

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
result_type std::linear_congruential_engine< _UIntType, __a, __c, __m >::modulus  [static], [constexpr]
```

The modulus.

multiplier

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
result_type std::linear_congruential_engine< _UIntType, __a, __c, __m >::multiplier  [static],
[constexpr]
```

The multiplier.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

5.629 __gnu_pbds::linear_probe_fn< Size_Type > Class Template Reference

```
#include <hash_policy.hpp>
```

Public Types

- typedef Size_Type **size_type**

Public Member Functions

- void **swap** (linear_probe_fn< Size_Type > &other)

Protected Member Functions

- size_type operator() (size_type i) const

5.629.1 Detailed Description

template<typename Size_Type = std::size_t>
class __gnu_pbds::linear_probe_fn< Size_Type >

A probe sequence policy using fixed increments.

5.629.2 Member Function Documentation

operator()()

```
template<typename Size_Type>
linear_probe_fn< Size_Type >::size_type __gnu_pbds::linear_probe_fn< Size_Type >::operator() (
            size_type i) const  [inline], [protected]
```

Returns the i-th offset from the hash value.

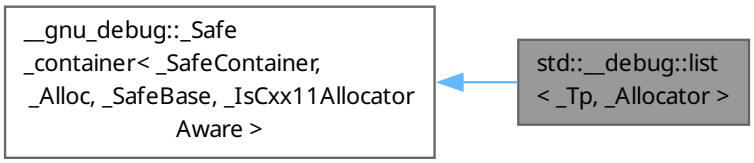
The documentation for this class was generated from the following file:

- hash_policy.hpp

5.630 std::__debug::list< _Tp, _Allocator > Class Template Reference

```
#include <list>
```

Inheritance diagram for std::__debug::list< _Tp, _Allocator >:



Public Types

- typedef _Allocator **allocator_type**
- typedef __gnu_debug::_Safe_iterator< _Base_const_iterator, list > **const_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**
- typedef std::reverse_iterator< const_iterator > **const_reverse_iterator**
- typedef _Base::difference_type **difference_type**
- typedef __gnu_debug::_Safe_iterator< _Base_iterator, list > **iterator**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef std::reverse_iterator< iterator > **reverse_iterator**
- typedef _Base::size_type **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **list** (_Base_ref __x)
- template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  list (_InputIterator __first, _InputIterator __last, const _Allocator &__a=_Allocator())
- **list** (const _Allocator &__a) noexcept
- **list** (const list &)=default
- **list** (const list &__x, const __type_identity_t< allocator_type > &__a)
- **list** (initializer_list< value_type > __l, const allocator_type &__a=allocator_type())
- **list** (list &&)=default

- **list** (list &&__x, const __type_identity_t< allocator_type > &__a) noexcept(std::is_nothrow_constructible< _Base, _Base, const allocator_type & >::value)
- **list** (size_type __n, const __type_identity_t< _Tp > &__value, const _Allocator &__a=_Allocator())
- **list** (size_type __n, const allocator_type &__a=allocator_type())
- const _Base & **_M_base** () const noexcept
- _Base & **_M_base** () noexcept
- template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  void **assign** (_InputIterator __first, _InputIterator __last)
- void **assign** (initializer_list< value_type > __l)
- void **assign** (size_type __n, const _Tp &__t)
- const_reference **back** () const noexcept
- reference **back** () noexcept
- const_iterator **begin** () const noexcept
- iterator **begin** () noexcept
- const_iterator **cbegin** () const noexcept
- const_iterator **cend** () const noexcept
- void **clear** () noexcept
- const_reverse_iterator **crbegin** () const noexcept
- const_reverse_iterator **crend** () const noexcept
- template<typename... _Args>
  iterator **emplace** (const_iterator __position, _Args &&... __args)
- const_iterator **end** () const noexcept
- iterator **end** () noexcept
- iterator **erase** (const_iterator __first, const_iterator __last) noexcept
- iterator **erase** (const_iterator __position) noexcept
- const_reference **front** () const noexcept
- reference **front** () noexcept
- iterator **insert** (const_iterator __p, initializer_list< value_type > __l)
- template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  iterator **insert** (const_iterator __position, _InputIterator __first, _InputIterator __last)
- iterator **insert** (const_iterator __position, _Tp &&__x)
- iterator **insert** (const_iterator __position, const _Tp &__x)
- iterator **insert** (const_iterator __position, size_type __n, const _Tp &__x)
- void **merge** (list &&__x)
- template<class _Compare>
  void **merge** (list &&__x, _Compare __comp)
- void **merge** (list &__x)
- template<typename _Compare>
  void **merge** (list &__x, _Compare __comp)
- list & **operator=** (const list &)=default
- list & **operator=** (initializer_list< value_type > __l)
- list & **operator=** (list &&)=default
- void **pop_back** () noexcept
- void **pop_front** () noexcept
- const_reverse_iterator **rbegin** () const noexcept
- reverse_iterator **rbegin** () noexcept
- __remove_return_type **remove** (const _Tp &__value)
- template<class _Predicate>
  __remove_return_type **remove_if** (_Predicate __pred)
- const_reverse_iterator **rend** () const noexcept
- reverse_iterator **rend** () noexcept

- void **resize** (size_type __sz)
- void **resize** (size_type __sz, const _Tp &__c)
- void **sort** ()
- template<typename _StrictWeakOrdering>
  void **sort** (_StrictWeakOrdering __pred)
- void **splice** (const_iterator __position, list &&__x) noexcept
- void **splice** (const_iterator __position, list &&__x, const_iterator __first, const_iterator __last) noexcept
- void **splice** (const_iterator __position, list &&__x, const_iterator __i) noexcept
- void **splice** (const_iterator __position, list &__x) noexcept
- void **splice** (const_iterator __position, list &__x, const_iterator __first, const_iterator __last) noexcept
- void **splice** (const_iterator __position, list &__x, const_iterator __i) noexcept
- void **swap** (list &__x) noexcept(/*conditional */)
- __remove_return_type **unique** ()
- template<class _BinaryPredicate>
  __remove_return_type **unique** (_BinaryPredicate __binary_pred)

Protected Member Functions

- constexpr void **_M_swap** (const _Safe_container &__x) const noexcept

Friends

- template<typename _ItT, typename _SeqT, typename _CatT>
  class **::__gnu_debug::_Safe_iterator**

5.630.1 Detailed Description

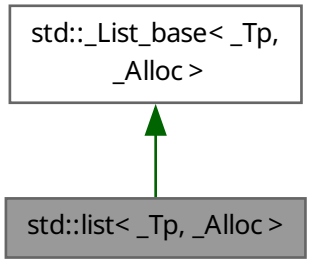
template<typename _Tp, typename _Allocator = std::allocator<_Tp>>
class std::__debug::list< _Tp, _Allocator >

Class std::list with safety/checking/debug instrumentation.
The documentation for this class was generated from the following file:

- debug/list

5.631 std::list< _Tp, _Alloc > Class Template Reference

`#include <stl_list.h>`
Inheritance diagram for std::list< _Tp, _Alloc >:



Public Types

- typedef _Alloc **allocator_type**
- typedef _Node_traits::_Const_iterator **const_iterator**
- typedef _Tp_alloc_traits::const_pointer **const_pointer**
- typedef _Tp_alloc_traits::const_reference **const_reference**
- typedef std::reverse_iterator< const_iterator > **const_reverse_iterator**
- typedef ptrdiff_t **difference_type**
- typedef _Node_traits::_Iterator **iterator**
- typedef _Tp_alloc_traits::pointer **pointer**
- typedef _Tp_alloc_traits::reference **reference**
- typedef std::reverse_iterator< iterator > **reverse_iterator**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- list ()=default
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  list (_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type())
- list (const allocator_type &__a) noexcept
- list (const list &__x)
- **list** (const list &__x, const __type_identity_t< allocator_type > &__a)
- list (initializer_list< value_type > __l, const allocator_type &__a=allocator_type())
- list (list &&)=default
- **list** (list &&__x, const __type_identity_t< allocator_type > &__a) noexcept(_Node_alloc_traits::_S_always_equal())
- list (size_type __n, const allocator_type &__a=allocator_type())
- list (size_type __n, const value_type &__value, const allocator_type &__a=allocator_type())
- ~list ()=default
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  void assign (_InputIterator __first, _InputIterator __last)
- void assign (initializer_list< value_type > __l)
- void assign (size_type __n, const value_type &__val)
- const_reference back () const noexcept
- reference back () noexcept
- const_iterator begin () const noexcept
- iterator begin () noexcept
- const_iterator cbegin () const noexcept
- const_iterator cend () const noexcept
- void clear () noexcept
- const_reverse_iterator crbegin () const noexcept
- const_reverse_iterator crend () const noexcept
- template<typename... _Args>
  iterator emplace (const_iterator __position, _Args &&... __args)
- template<typename... _Args>
  reference **emplace_back** (_Args &&... __args)
- template<typename... _Args>
  reference **emplace_front** (_Args &&... __args)
- bool empty () const noexcept
- const_iterator end () const noexcept
- iterator end () noexcept

- iterator erase (const_iterator __first, const_iterator __last) noexcept
- iterator erase (const_iterator __position) noexcept
- const_reference front () const noexcept
- reference front () noexcept
- allocator_type get_allocator () const noexcept
- iterator insert (const_iterator __p, initializer_list< value_type > __l)
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  iterator insert (const_iterator __position, _InputIterator __first, _InputIterator __last)
- iterator insert (const_iterator __position, size_type __n, const value_type &__x)
- size_type max_size () const noexcept
- void merge (list &&__x)
- template<typename _StrictWeakOrdering>
  void merge (list &&__x, _StrictWeakOrdering __comp)
- void **merge** (list &__x)
- template<typename _StrictWeakOrdering>
  void **merge** (list &__x, _StrictWeakOrdering __comp)
- list & operator= (const list &__x)
- list & operator= (initializer_list< value_type > __l)
- list & operator= (list &&__x) noexcept(_Node_alloc_traits::_S_nothrow_move())
- void pop_back () noexcept
- void pop_front () noexcept
- void push_back (const value_type &__x)
- void **push_back** (value_type &&__x)
- void push_front (const value_type &__x)
- void **push_front** (value_type &&__x)
- const_reverse_iterator rbegin () const noexcept
- reverse_iterator rbegin () noexcept
- __remove_return_type remove (const _Tp &__value)
- template<typename _Predicate>
  __remove_return_type remove_if (_Predicate)
- const_reverse_iterator rend () const noexcept
- reverse_iterator rend () noexcept
- void resize (size_type __new_size)
- void resize (size_type __new_size, const value_type &__x)
- void reverse () noexcept
- size_type size () const noexcept
- void sort ()
- template<typename _StrictWeakOrdering>
  void sort (_StrictWeakOrdering)
- void splice (const_iterator __position, list &&__x) noexcept
- void splice (const_iterator __position, list &&__x, const_iterator __first, const_iterator __last) noexcept
- void **splice** (const_iterator __position, list &__x) noexcept
- void splice (const_iterator __position, list &__x, const_iterator __first, const_iterator __last) noexcept
- void swap (list &__x) noexcept
- __remove_return_type unique ()
- template<typename _BinaryPredicate>
  __remove_return_type unique (_BinaryPredicate)

- iterator insert (const_iterator __position, const value_type &__x)
- iterator insert (const_iterator __position, value_type &&__x)

- void splice (const_iterator __position, list &&__x, const_iterator __i) noexcept
- void splice (const_iterator __position, list &__x, const_iterator __i) noexcept

Protected Types

- typedef _Node_alloc_traits::pointer **_Node_ptr**

Protected Member Functions

- template<typename _InputIterator>
  void **_M_assign_dispatch** (_InputIterator __first, _InputIterator __last, __false_type)
- template<typename _Integer>
  void **_M_assign_dispatch** (_Integer __n, _Integer __val, __true_type)
- void **_M_check_equal_allocators** (const list &__x) noexcept
- void **_M_clear** () noexcept
- template<typename... _Args>
  _Node_ptr _M_create_node (_Args &&... __args)
- void **_M_dec_size** (size_t __n)
- void **_M_default_append** (size_type __n)
- void **_M_default_initialize** (size_type __n)
- void **_M_destroy_node** (_Node_ptr __p)
- size_t **_M_distance** (const __detail::_List_node_base *__first, const __detail::_List_node_base *__last) const
- void **_M_erase** (iterator __position) noexcept
- void **_M_fill_assign** (size_type __n, const value_type &__val)
- void **_M_fill_initialize** (size_type __n, const value_type &__x)
- _Node_alloc_traits::pointer **_M_get_node** ()
- const _Node_alloc_type & **_M_get_Node_allocator** () const noexcept
- _Node_alloc_type & **_M_get_Node_allocator** () noexcept
- size_t **_M_get_size** () const
- void **_M_inc_size** (size_t __n)
- void **_M_init** () noexcept
- template<typename _InputIterator>
  void **_M_initialize_dispatch** (_InputIterator __first, _InputIterator __last, __false_type)
- template<typename _Integer>
  void **_M_initialize_dispatch** (_Integer __n, _Integer __x, __true_type)
- template<typename... _Args>
  void **_M_insert** (iterator __position, _Args &&... __args)
- void **_M_move_assign** (list &&__x, false_type)
- void **_M_move_assign** (list &&__x, true_type) noexcept
- void **_M_move_nodes** (_List_base &&__x)
- size_t **_M_node_count** () const
- void **_M_put_node** (_Node_ptr __p) noexcept
- const_iterator **_M_resize_pos** (size_type &__new_size) const
- void **_M_set_size** (size_t __n)
- void **_M_transfer** (iterator __position, iterator __first, iterator __last)

Static Protected Member Functions

- static size_t **_S_distance** (const __detail::_List_node_base *__first, const __detail::_List_node_base *__last)
- static size_t **_S_distance** (const_iterator __first, const_iterator __last)

Protected Attributes

- _List_impl **_M_impl**

5.631.1 Detailed Description

template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
class std::list< _Tp, _Alloc >

A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.

Template Parameters

_Tp	Type of element.
_Alloc	Allocator type, defaults to allocator<_Tp>.

Meets the requirements of a `container`, a `reversible container`, and a `sequence`, including the `optional sequence requirements` with the exception of `at` and `operator[]`.
This is a *doubly linked* list. Traversal up and down the list requires linear time, but adding and removing elements (or *nodes*) is done in constant time, regardless of where the change takes place. Unlike std::vector and std::deque, random-access iterators are not provided, so subscripting ([]) access is not allowed. For algorithms which only need sequential access, this lack makes no difference.
Also unlike the other standard containers, std::list provides specialized algorithms unique to linked lists, such as splicing, sorting, and in-place reversal.
A couple points on memory allocation for list<Tp>:
First, we never actually allocate a Tp, we allocate List_node<Tp>'s and trust [20.1.5]/4 to DTRT. This is to ensure that after elements from list<X,Alloc1> are spliced into list<X,Alloc2>, destroying the memory of the second list is a valid operation, i.e., Alloc1 giveth and Alloc2 taketh away.
Second, a list conceptually represented as

```
A <---> B <---> C <---> D
```

is actually circular; a link exists between A and D. The list class holds (as its only data member) a private list::iterator pointing to *D*, not to *A!* To get to the head of the list, we start at the tail and move forward by one. When this member iterator's next/previous pointers refer to itself, the list is empty.

5.631.2 Constructor & Destructor Documentation

list() [1/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list< _Tp, _Alloc >::list ()  [default]
```

Creates a list with no elements.
Referenced by erase(), insert(), insert(), insert(), merge(), merge(), operator=(), remove(), remove_if(), unique(), and unique().

list() [2/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list< _Tp, _Alloc >::list (
            const allocator_type & __a)  [inline], [explicit], [noexcept]
```

Creates a list with no elements.

Parameters

_↩ _a	An allocator object.

list() [3/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list< _Tp, _Alloc >::list (
            size_type __n,
            const allocator_type & __a = allocator_type())  [inline], [explicit]
```

Creates a list with default constructed elements.

Parameters

_↩ _n	The number of elements to initially create.
_↩ _a	An allocator object.

This constructor fills the list with *__n* default constructed elements.

list() [4/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list< _Tp, _Alloc >::list (
            size_type __n,
            const value_type & __value,
            const allocator_type & __a = allocator_type())  [inline]
```

Creates a list with copies of an exemplar element.

Parameters

__n	The number of elements to initially create.
__value	An element to copy.
__a	An allocator object.

This constructor fills the list with *__n* copies of *__value*.

list() [5/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list< _Tp, _Alloc >::list (
            const list< _Tp, _Alloc > & __x)  [inline]
```

List copy constructor.

Parameters

_↩ _x	A list of identical element and allocator types.

The newly-created list uses a copy of the allocation object used by *__x* (unless the allocator traits dictate a different object).

list() [6/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list< _Tp, _Alloc >::list (
            list< _Tp, _Alloc > && )  [default]
```

List move constructor.
The newly-created list contains the exact contents of the moved instance. The contents of the moved instance are a valid, but unspecified list.

list() [7/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list< _Tp, _Alloc >::list (
            initializer_list< value_type > __l,
            const allocator_type & __a = allocator_type())  [inline]
```

Builds a list from an initializer_list.

Parameters

_↩ _l	An initializer_list of value_type.
_↩ _a	An allocator object.

Create a list consisting of copies of the elements in the initializer_list *__l*. This is linear in __l.size().

list() [8/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
std::list< _Tp, _Alloc >::list (
            _InputIterator __first,
            _InputIterator __last,
            const allocator_type & __a = allocator_type())  [inline]
```

Builds a list from a range.

Parameters

__first	An input iterator.
__last	An input iterator.
__a	An allocator object.

Create a list consisting of copies of the elements from [*__first*,*__last*). This is linear in N (where N is distance(*__first*,↩ *__last*)).

~list()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list< _Tp, _Alloc >::~list ()  [default]
```

No explicit dtor needed as the _Base dtor takes care of things. The _Base dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

5.631.3 Member Function Documentation

_M_create_node()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename...  _Args>
```

```
_Node_ptr std::list< _Tp, _Alloc >::_M_create_node (
            _Args &&... __args)  [inline], [protected]
```

Parameters

__args	An instance of user data.

Allocates space for a new node and constructs a copy of *__args* in it.
Referenced by emplace(), and insert().

assign() [1/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
void std::list< _Tp, _Alloc >::assign (
            _InputIterator __first,
            _InputIterator __last)  [inline]
```

Assigns a range to a list.

Parameters

__first	An input iterator.
__last	An input iterator.

This function fills a list with copies of the elements in the range [*__first*,*__last*).
Note that the assignment completely changes the list and that the resulting list's size is the same as the number of elements assigned.

assign() [2/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::assign (
            initializer_list< value_type > __l)  [inline]
```

Assigns an initializer_list to a list.

Parameters

↩ _↩ ↩ _↩ *l*	An initializer_list of value_type.

Replace the contents of the list with copies of the elements in the initializer_list *__l*. This is linear in __l.size().

assign() [3/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::assign (
            size_type __n,
            const value_type & __val)  [inline]
```

Assigns a given value to a list.

Parameters

__n	Number of elements to be assigned.
__val	Value to be assigned.

This function fills a list with __*n* copies of the given value. Note that the assignment completely changes the list and that the resulting list's size is the same as the number of elements assigned.

back() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::list< _Tp, _Alloc >::back () const  [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) reference to the data at the last element of the list.

back() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::list< _Tp, _Alloc >::back ()  [inline], [nodiscard], [noexcept]
```

Returns a read/write reference to the data at the last element of the list.

begin() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::list< _Tp, _Alloc >::begin () const  [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the list. Iteration is done in ordinary element order.

begin() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::list< _Tp, _Alloc >::begin ()  [inline], [nodiscard], [noexcept]
```

Returns a read/write iterator that points to the first element in the list. Iteration is done in ordinary element order.
Referenced by std::list< _Tp, polymorphic_allocator< _Tp > >::list(), insert(), insert(), merge(), merge(), std::operator<=>(), operator=(), std::list< _Tp, polymorphic_allocator< _Tp > >::operator=(), remove(), remove_if(), std::list< _Tp, polymorphic_allocator< _ unique(), and unique().

cbegin()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::list< _Tp, _Alloc >::cbegin () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the list. Iteration is done in ordinary element order.

cend()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::list< _Tp, _Alloc >::cend () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the list. Iteration is done in ordinary element order.

clear()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::clear ()  [inline], [noexcept]
```

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.
Referenced by operator=().

crbegin()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::list< _Tp, _Alloc >::crbegin () const  [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the list. Iteration is done in reverse element order.

crend()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::list< _Tp, _Alloc >::crend () const  [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first element in the list. Iteration is done in reverse element order.

emplace()

```
template<typename _Tp, typename _Alloc>
template<typename...  _Args>
list< _Tp, _Alloc >::iterator list::emplace (
            const_iterator __position,
            _Args &&...  __args)
```

Constructs object in list before specified iterator.

Parameters

__position	A const_iterator into the list.
__args	Arguments.

Returns

An iterator that points to the inserted data.

This function will insert an object of type T constructed with T(std::forward<Args>(args)...) before the specified location. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.
References _M_create_node(), and std::forward().

empty()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
bool std::list< _Tp, _Alloc >::empty () const  [inline], [nodiscard], [noexcept]
```

Returns true if the list is empty. (Thus begin() would equal end().)
Referenced by insert(), and std::list< _Tp, polymorphic_allocator< _Tp > >::splice().

end() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::list< _Tp, _Alloc >::end () const  [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the list. Iteration is done in ordinary element order.

end() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::list< _Tp, _Alloc >::end ()  [inline], [nodiscard], [noexcept]
```

Returns a read/write iterator that points one past the last element in the list. Iteration is done in ordinary element order.
Referenced by std::list< _Tp, polymorphic_allocator< _Tp > >::list(), merge(), merge(), std::operator<=>(), operator=(), std::list< _Tp, polymorphic_allocator< _Tp > >::operator=(), and std::list< _Tp, polymorphic_allocator< _Tp > >::splice().

erase() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::list< _Tp, _Alloc >::erase (
            const_iterator __first,
            const_iterator __last)  [inline], [noexcept]
```

Remove a range of elements.

Parameters

__first	Iterator pointing to the first element to be erased.
__last	Iterator pointing to one past the last element to be erased.

Returns

An iterator pointing to the element pointed to by *last* prior to erasing (or end()).

This function will erase the elements in the range [first,last) and shorten the list accordingly.
This operation is linear time in the size of the range and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [2/2]

```
template<typename _Tp, typename _Alloc>
list< _Tp, _Alloc >::iterator list::erase (
            const_iterator __position)  [noexcept]
```

Remove element at given position.

Parameters

__position	Iterator pointing to element to be erased.

Returns

An iterator pointing to the next element (or end()).

This function will erase the element at the given position and thus shorten the list by one.
Due to the nature of a list this operation can be done in constant time, and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.
References list(), and erase().
Referenced by erase(), resize(), and resize().

front() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::list< _Tp, _Alloc >::front () const  [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) reference to the data at the first element of the list.

front() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::list< _Tp, _Alloc >::front ()  [inline], [nodiscard], [noexcept]
```

Returns a read/write reference to the data at the first element of the list.

get_allocator()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
allocator_type std::list< _Tp, _Alloc >::get_allocator () const  [inline], [noexcept]
```

Get a copy of the memory allocation object.
Referenced by insert(), insert(), remove(), remove_if(), unique(), and unique().

insert() [1/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::list< _Tp, _Alloc >::insert (
            const_iterator __p,
            initializer_list< value_type > __l)  [inline]
```

Inserts the contents of an initializer_list into list before specified const_iterator.

Parameters

_↩ _p	A const_iterator into the list.
_↩ _l	An initializer_list of value_type.

Returns

An iterator pointing to the first element inserted (or __p).

This function will insert copies of the data in the initializer_list __l into the list before the location specified by __p.
This operation is linear in the number of elements inserted and does not invalidate iterators and references.

insert() [2/5]

```
template<typename _Tp, typename _Alloc>
template<typename _InputIterator, typename>
list< _Tp, _Alloc >::iterator list::insert (
            const_iterator __position,
            _InputIterator __first,
            _InputIterator __last)
```

Inserts a range into the list.

Parameters

__position	An iterator into the list.
__first	An input iterator.
__last	An input iterator.

Returns

Since C++11, an iterator pointing to the first element inserted (or __position). In C++98 this returns void.

This function will insert copies of the data in the range [__first,__last) into the list before the location specified by __position.
This operation is linear in the number of elements inserted and does not invalidate iterators and references.
References list(), begin(), empty(), get_allocator(), and splice().

insert() [3/5]

```
template<typename _Tp, typename _Alloc>
list< _Tp, _Alloc >::iterator list::insert (
            const_iterator __position,
            const value_type & __x)
```

Inserts given value into list before specified iterator.

Parameters

__position	An iterator into the list.
__x	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.
References list(), _M_create_node(), and insert().
Referenced by insert(), and resize().

insert() [4/5]

```
template<typename _Tp, typename _Alloc>
list< _Tp, _Alloc >::iterator list::insert (
            const_iterator __position,
            size_type __n,
            const value_type & __x)
```

Inserts a number of copies of given data into the list.

Parameters

__position	An iterator into the list.
__n	Number of elements to be inserted.
__x	Data to be inserted.

Returns

Since C++11, an iterator pointing to the first element inserted (or `__position`). In C++98 this returns void.

This function will insert a specified number of copies of the given data before the location specified by `__position`. This operation is linear in the number of elements inserted and does not invalidate iterators and references.
References list(), begin(), get_allocator(), and splice().

insert() [5/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::list< _Tp, _Alloc >::insert (
            const_iterator __position,
            value_type && __x)  [inline]
```

Inserts given value into list before specified iterator.

Parameters

__position	An iterator into the list.
__x	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

max_size()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::list< _Tp, _Alloc >::max_size () const  [inline], [nodiscard], [noexcept]
```

Returns the size() of the largest possible list.

merge() [1/2]

```
template<typename _Tp, typename _Alloc>
void list::merge (
            list< _Tp, _Alloc > && __x)
```

Merge sorted lists.

Parameters

_↩ _x	Sorted list to merge.

Assumes that both *__x* and this list are sorted according to operator<(). Merges elements of *__x* into this list in sorted order, leaving *__x* empty when complete. Elements in this list precede elements in *__x* that are equal.
References list(), std::__addressof(), std::begin(), begin(), std::end(), end(), and merge().
Referenced by merge(), and merge().

merge() [2/2]

```
template<typename _Tp, typename _Alloc>
template<typename _StrictWeakOrdering>
void list::merge (
            list< _Tp, _Alloc > && __x,
            _StrictWeakOrdering __comp)
```

Merge sorted lists according to comparison function.

Template Parameters

_StrictWeakOrdering	Comparison function defining sort order.

Parameters

__x	Sorted list to merge.
__comp	Comparison functor.

Assumes that both __x and this list are sorted according to StrictWeakOrdering. Merges elements of __x into this list in sorted order, leaving __x empty when complete. Elements in this list precede elements in __x that are equivalent according to StrictWeakOrdering().
References list(), std::__addressof(), std::begin(), begin(), std::end(), end(), and merge().

operator=() [1/3]

```
template<typename _Tp, typename _Alloc>
list< _Tp, _Alloc > & list::operator= (
            const list< _Tp, _Alloc > & __x)
```

List assignment operator.

Parameters

_↩ _x	A list of identical element and allocator types.

All the elements of __x are copied.
Whether the allocator is copied depends on the allocator traits.
References list(), std::__addressof(), begin(), clear(), and end().

operator=() [2/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
list & std::list< _Tp, _Alloc >::operator= (
            initializer_list< value_type > __l)  [inline]
```

List initializer list assignment operator.

Parameters

↩ _↩ ↩ _↩ l	An initializer_list of value_type.

Replace the contents of the list with copies of the elements in the initializer_list __l. This is linear in l.size().

operator=() [3/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
list & std::list< _Tp, _Alloc >::operator= (
            list< _Tp, _Alloc > && __x)  [inline], [noexcept]
```

List move assignment operator.

Parameters

_↩ _x	A list of identical element and allocator types.

The contents of __x are moved into this list (without copying).
Afterwards __x is a valid, but unspecified list
Whether the allocator is moved depends on the allocator traits.

pop_back()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::pop_back ()  [inline], [noexcept]
```

Removes last element.

This is a typical stack operation. It shrinks the list by one. Due to the nature of a list this operation can be done in constant time, and only invalidates iterators/references to the element being removed.

Note that no data is returned, and if the last element's data is needed, it should be retrieved before pop_back() is called.

pop_front()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::pop_front ()  [inline], [noexcept]
```

Removes first element.

This is a typical stack operation. It shrinks the list by one. Due to the nature of a list this operation can be done in constant time, and only invalidates iterators/references to the element being removed.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before pop_front() is called.

push_back()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::push_back (
            const value_type & __x)  [inline]
```

Add data to the end of the list.

Parameters

_↩ _x	Data to be added.

This is a typical stack operation. The function creates an element at the end of the list and assigns the given data to it. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

push_front()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::push_front (
            const value_type & __x)  [inline]
```

Add data to the front of the list.

Parameters

_↩ _x	Data to be added.

This is a typical stack operation. The function creates an element at the front of the list and assigns the given data to it. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

rbegin() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::list< _Tp, _Alloc >::rbegin () const  [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the list. Iteration is done in reverse element order.

rbegin() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::list< _Tp, _Alloc >::rbegin ()  [inline], [nodiscard], [noexcept]
```

Returns a read/write reverse iterator that points to the last element in the list. Iteration is done in reverse element order.

remove()

```
template<typename _Tp, typename _Alloc>
list< _Tp, _Alloc >::__remove_return_type list::remove (
            const _Tp & __value)
```

Remove all elements equal to value.

Parameters

__value	The value to remove.

Removes every element in the list equal to *value*. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.
References list(), std::begin(), begin(), std::end(), get_allocator(), size(), and splice().

remove_if()

```
template<typename _Tp, typename _Alloc>
template<typename _Predicate>
list< _Tp, _Alloc >::__remove_return_type list::remove_if (
            _Predicate __pred)
```

Remove all elements satisfying a predicate.

Template Parameters

_Predicate	Unary predicate function or object.

Removes every element in the list for which the predicate returns true. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.
References list(), std::begin(), begin(), std::end(), get_allocator(), size(), and splice().

rend() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::list< _Tp, _Alloc >::rend () const  [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first element in the list. Iteration is done in reverse element order.

rend() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::list< _Tp, _Alloc >::rend ()  [inline], [nodiscard], [noexcept]
```

Returns a read/write reverse iterator that points to one before the first element in the list. Iteration is done in reverse element order.

resize() [1/2]

```
template<typename _Tp, typename _Alloc>
void list::resize (
            size_type __new_size)
```

Resizes the list to the specified number of elements.

Parameters

__new_size	Number of elements the list should contain.

This function will resize the list to the specified number of elements. If the number is smaller than the list's current size the list is truncated, otherwise default constructed elements are appended.
References std::end(), and erase().

resize() [2/2]

```
template<typename _Tp, typename _Alloc>
void list::resize (
            size_type __new_size,
            const value_type & __x)
```

Resizes the list to the specified number of elements.

Parameters

__new_size	Number of elements the list should contain.
__x	Data with which new elements should be populated.

This function will resize the list to the specified number of elements. If the number is smaller than the list's current size the list is truncated, otherwise the list is extended and new elements are populated with given data.
References std::end(), erase(), and insert().

reverse()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::reverse ()  [inline], [noexcept]
```

Reverse the elements in list.
Reverse the order of elements in the list in linear time.

size()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::list< _Tp, _Alloc >::size () const  [inline], [nodiscard], [noexcept]
```

Returns the number of elements in the list.
Referenced by std::operator<=>(), remove(), remove_if(), unique(), and unique().

sort() [1/2]

```
template<typename _Tp, typename _Alloc>
void list::sort ()
```

Sort the elements.
Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.
References std::begin(), std::empty(), and std::end().

sort() [2/2]

```
template<typename _Tp, typename _Alloc>
template<typename _StrictWeakOrdering>
void list::sort (
            _StrictWeakOrdering __comp)
```

Sort the elements according to comparison function.
Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.
References std::begin(), std::empty(), and std::end().

splice() [1/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::splice (
            const_iterator __position,
            list< _Tp, _Alloc > && __x)  [inline], [noexcept]
```

Insert contents of another list.

Parameters

__position	Iterator referencing the element to insert before.
__x	Source list.

The elements of *__x* are inserted in constant time in front of the element referenced by *__position*. *__x* becomes an empty list.
Requires this != *__x*.
Referenced by insert(), insert(), remove(), remove_if(), unique(), and unique().

splice() [2/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::splice (
            const_iterator __position,
            list< _Tp, _Alloc > && __x,
            const_iterator __first,
            const_iterator __last)  [inline], [noexcept]
```

Insert range from another list.

Parameters

__position	Iterator referencing the element to insert before.
__x	Source list.
__first	Iterator referencing the start of range in x.
__last	Iterator referencing the end of range in x.

Removes elements in the range [__first,__last) and inserts them before *__position* in constant time.
Undefined if *__position* is in [__first,__last).

splice() [3/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::splice (
            const_iterator __position,
            list< _Tp, _Alloc > && __x,
            const_iterator __i)  [inline], [noexcept]
```

Insert element from another list.

Parameters

__position	Iterator referencing the element to insert before.
__x	Source list.
__i	Iterator referencing the element to move.

Removes the element in list *__x* referenced by *__i* and inserts it into the current list before *__position*.

splice() [4/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::splice (
            const_iterator __position,
            list< _Tp, _Alloc > & __x,
            const_iterator __first,
            const_iterator __last)  [inline], [noexcept]
```

Insert range from another list.

Parameters

__position	Const_iterator referencing the element to insert before.
__x	Source list.
__first	Const_iterator referencing the start of range in x.
__last	Const_iterator referencing the end of range in x.

Removes elements in the range [__first,__last) and inserts them before *__position* in constant time. Undefined if *__position* is in [__first,__last).

splice() [5/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::splice (
            const_iterator __position,
            list< _Tp, _Alloc > & __x,
            const_iterator __i)  [inline], [noexcept]
```

Insert element from another list.

Parameters

__position	Iterator referencing the element to insert before.
__x	Source list.
__i	Iterator referencing the element to move.

Removes the element in list *__x* referenced by *__i* and inserts it into the current list before *__position*.

swap()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::swap (
            list< _Tp, _Alloc > & __x)  [inline], [noexcept]
```

Swaps data with another list.

Parameters

_↩ _x	A list of the same element and allocator types.

This exchanges the elements between two lists in constant time. Note that the global std::swap() function is specialized such that std::swap(l1,l2) will feed to this function.
Whether the allocators are swapped depends on the allocator traits.

unique() [1/2]

```
template<typename _Tp, typename _Alloc>
list< _Tp, _Alloc >::__remove_return_type list::unique ()
```

Remove consecutive duplicate elements.
For each consecutive set of elements with the same value, remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.
References list(), std::begin(), begin(), std::end(), get_allocator(), size(), and splice().

unique() [2/2]

```
template<typename _Tp, typename _Alloc>
template<typename _BinaryPredicate>
list< _Tp, _Alloc >::__remove_return_type list::unique (
            _BinaryPredicate __binary_pred)
```

Remove consecutive elements satisfying a predicate.

Template Parameters

_BinaryPredicate	Binary predicate function or object.

For each consecutive set of elements [first,last) that satisfy predicate(first,i) where i is an iterator in [first,last), remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.
References list(), std::begin(), begin(), std::end(), get_allocator(), size(), and splice().
The documentation for this class was generated from the following files:

- stl_list.h
- list.tcc

5.632 __gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc > Class Template Reference

```
#include <assoc_container.hpp>
```

Public Types

- typedef list_update_tag **container_category**
- typedef Eq_Fn **eq_fn**
- typedef Update_Policy **update_policy**

Public Member Functions

- **list_update** (const list_update &other)
- template<typename It>
  list_update (It first, It last)
- list_update & **operator=** (const list_update &other)
- void **swap** (list_update &other)

5.632.1 Detailed Description

template<typename Key, typename Mapped, class Eq_Fn = typename detail::default_eq_fn<Key>::type, class Update_Policy = detail::default_update_policy::type, class _Alloc = std::allocator<char>>
class __gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >

A list-update based associative container.

Template Parameters

Key	Key type.
Mapped	Map type.
Eq_Fn	Equal functor.
Update_Policy	Update policy, determines when an element will be moved to the front of the list.
_Alloc	Allocator type.

Base is detail::lu_map.

5.632.2 Constructor & Destructor Documentation

list_update()

```
template<typename Key, typename Mapped, class Eq_Fn = typename detail::default_eq_fn<Key>::type,
class Update_Policy = detail::default_update_policy::type, class _Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >::list_update (
            It first,
            It last)  [inline]
```

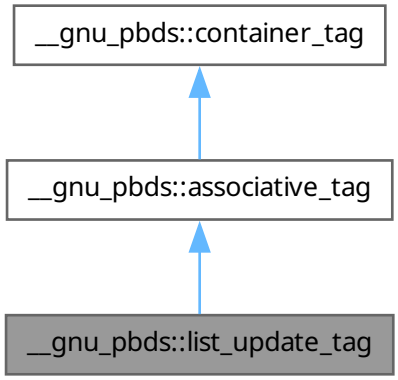
Constructor taking __iterators to a range of value_types. The value_types between first_it and last_it will be inserted into the container object.
The documentation for this class was generated from the following file:

- assoc_container.hpp

5.633 __gnu_pbds::list_update_tag Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for __gnu_pbds::list_update_tag:



5.633.1 Detailed Description

List-update.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

5.634 std::locale Class Reference

`#include <locale_classes.h>`

Classes

- class facet
- class id

Public Types

- typedef int category

Public Member Functions

- locale () throw ()
- locale (const char *__s)
- locale (const locale &__base, const char *__s, category __cat)
- locale (const locale &__base, const locale &__add, category __cat)
- locale (const locale &__base, const std::string &__s, category __cat)
- locale (const locale &__other) throw ()
- template<typename _Facet>
  locale (const locale &__other, _Facet *__f)
- locale (const std::string &__s)
- ~locale () throw ()

- template<typename _Facet>
  locale combine (const locale &__other) const
- string name () const
- template<typename _Char, typename _Traits, typename _Alloc>
  bool operator() (const basic_string< _Char, _Traits, _Alloc > &__s1, const basic_string< _Char, _Traits, _Alloc > &__s2) const
- template<typename _CharT, typename _Traits, typename _Alloc>
  bool **operator()** (const basic_string< _CharT, _Traits, _Alloc > &__s1, const basic_string< _CharT, _Traits, _↩ Alloc > &__s2) const
- const locale & operator= (const locale &__other) throw ()
- bool operator== (const locale &__other) const throw ()

Static Public Member Functions

- static const locale & classic ()
- static locale global (const locale &__loc)

Static Public Attributes

- static const category none
- static const category ctype
- static const category numeric
- static const category collate
- static const category time
- static const category monetary
- static const category messages
- static const category all

Friends

- template<typename _Facet>
  const _Facet * **__try_use_facet** (const locale &__loc)
- template<typename _Cache>
  struct **__use_cache**
- class **_Impl**
- class **facet**
- template<typename _Facet>
  bool has_facet (const locale &__loc) throw ()
- template<typename _Facet>
  const _Facet & use_facet (const locale &__loc)

5.634.1 Detailed Description

Container class for localization functionality.
The locale class is first a class wrapper for C library locales. It is also an extensible container for user-defined localization. A locale is a collection of facets that implement various localization features such as money, time, and number printing. Constructing C++ locales does not change the C library locale.
This library supports efficient construction and copying of locales through a reference counting implementation of the locale class.

5.634.2 Member Typedef Documentation

category

```
typedef int std::locale::category
```

Definition of locale::category.

5.634.3 Constructor & Destructor Documentation

locale() [1/8]

```
std::locale::locale () throw ( )
```

Default constructor.

Constructs a copy of the global locale. If no locale has been explicitly set, this is the C locale.

Referenced by locale(), locale(), locale(), locale(), locale(), locale(), locale(), ~locale(), combine(), global(), has_facet, operator()(), operator=(), operator==(), and use_facet.

locale() [2/8]

```
std::locale::locale (
            const locale & __other) throw ( )
```

Copy constructor.

Constructs a copy of *other*.

Parameters

__other	The locale to copy.

References locale().

locale() [3/8]

```
std::locale::locale (
            const char * __s)  [explicit]
```

Named locale constructor.

Constructs a copy of the named C library locale.

Parameters

__s	Name of the locale to construct.

Exceptions

std::runtime_error	if __s is null or an undefined locale.

References locale().

locale() [4/8]

```
std::locale::locale (
            const locale & __base,
            const char * __s,
            category __cat)
```

Construct locale with facets from another locale.
Constructs a copy of the locale *base*. The facets specified by *cat* are replaced with those from the locale named by *s*. If base is named, this locale instance will also be named.

Parameters

__base	The locale to copy.
__s	Name of the locale to use facets from.
__cat	Set of categories defining the facets to use from __s.

Exceptions

std::runtime_error	if __s is null or an undefined locale.

References locale().

locale() [5/8]

```
std::locale::locale (
            const std::string & __s)  [inline], [explicit]
```

Named locale constructor.
Constructs a copy of the named C library locale.

Parameters

_↩ _s	Name of the locale to construct.

Exceptions

std::runtime_error	if __s is an undefined locale.

References locale().

locale() [6/8]

```
std::locale::locale (
            const locale & __base,
            const std::string & __s,
            category __cat)  [inline]
```

Construct locale with facets from another locale.
Constructs a copy of the locale *base*. The facets specified by *cat* are replaced with those from the locale named by *s*. If base is named, this locale instance will also be named.

Parameters

__base	The locale to copy.
__s	Name of the locale to use facets from.
__cat	Set of categories defining the facets to use from __s.

Exceptions

std::runtime_error	if __s is an undefined locale.

References locale().

locale() [7/8]

```
std::locale::locale (
            const locale & __base,
            const locale & __add,
            category __cat)
```

Construct locale with facets from another locale.
Constructs a copy of the locale *base*. The facets specified by *cat* are replaced with those from the locale *add*. If *base* and *add* are named, this locale instance will also be named.

Parameters

__base	The locale to copy.
__add	The locale to use facets from.
__cat	Set of categories defining the facets to use from add.

References locale().

locale() [8/8]

```
template<typename _Facet>
std::locale::locale (
            const locale & __other,
            _Facet * __f)
```

Construct locale with another facet.
Constructs a copy of the locale *__other*. The facet *__f* is added to *__other*, replacing an existing facet of type Facet if there is one. If *__f* is null, this locale is a copy of *__other*.

Parameters

__other	The locale to copy.
__f	The facet to add in.

References locale().

~locale()

```
std::locale::~locale () throw ( )
```

Locale destructor.
References locale().

5.634.4 Member Function Documentation

classic()

```
static const locale & std::locale::classic ()  [static], [nodiscard]
```

Return reference to the C locale.

combine()

```
template<typename _Facet>
locale std::locale::combine (
            const locale & __other) const  [nodiscard]
```

Construct locale with another facet.
Constructs and returns a new copy of this locale. Adds or replaces an existing facet of type Facet from the locale *other* into the new locale.

Template Parameters

_Facet	The facet type to copy from other

Parameters

__other	The locale to copy from.

Returns

Newly constructed locale.

Exceptions

std::runtime_error	if __other has no facet of type _Facet.

References locale().
Referenced by operator=().

global()

```
static locale std::locale::global (
            const locale & __loc)  [static]
```

Set global locale.
This function sets the global locale to the argument and returns a copy of the previous global locale. If the argument has a name, it will also call std::setlocale(LC_ALL, loc.name()).

Parameters

__loc	The new locale to make global.

Returns

Copy of the old global locale.

References locale().

name()

```
string std::locale::name () const  [nodiscard]
```

Return locale name.

Returns

Locale name or "*" if unnamed.

References name().
Referenced by name().

operator()()

```
template<typename _Char, typename _Traits, typename _Alloc>
bool std::locale::operator() (
            const basic_string< _Char, _Traits, _Alloc > & __s1,
            const basic_string< _Char, _Traits, _Alloc > & __s2) const  [nodiscard]
```

Compare two strings according to collate.

Template operator to compare two strings using the compare function of the collate facet in this locale. One use is to provide the locale to the sort function. For example, a vector v of strings could be sorted according to locale loc by doing:

```
std::sort(v.begin(), v.end(), loc);
```

Parameters

__s1	First string to compare.
__s2	Second string to compare.

Returns

True if collate<_Char> facet compares __s1 < __s2, else false.

References locale().

operator=()

```
const locale & std::locale::operator= (
            const locale & __other) throw ( )
```

Assignment operator.

Set this locale to be a copy of *other*.

Parameters

__other	The locale to copy.

Returns

A reference to this locale.

References locale(), and combine().

operator==()

```
bool std::locale::operator== (
            const locale & __other) const throw ( )   [nodiscard]
```

Locale equality.

Parameters

__other	The locale to compare against.

Returns

True if other and this refer to the same locale instance, are copies, or have the same name. False otherwise.

References locale(), and operator==().

Referenced by operator==().

5.634.5 Friends And Related Symbol Documentation

has_facet

```
template<typename _Facet>
bool has_facet (
            const locale & __loc) throw ( )   [friend]
```

Test for the presence of a facet.
has_facet tests the locale argument for the presence of the facet type provided as the template parameter. Facets derived from the facet parameter will also return true.

Template Parameters

_Facet	The facet type to test the presence of.

Parameters

__loc	The locale to test.

Returns

true if __loc contains a facet of type _Facet, else false.

References locale().

use_facet

```
template<typename _Facet>
const _Facet & use_facet (
            const locale & __loc)  [friend]
```

Return a facet.
use_facet looks for and returns a reference to a facet of type Facet where Facet is the template parameter. If has_facet(locale) is true, there is a suitable facet to return. It throws std::bad_cast if the locale doesn't contain a facet of type Facet.

Template Parameters

_Facet	The facet type to access.

Parameters

__loc	The locale to use.

Returns

Reference to facet of type Facet.

Exceptions

std::bad_cast	if __loc doesn't contain a facet of type _Facet.

References locale().

5.634.6 Member Data Documentation

all

```
const category std::locale::all  [static]
```

Category values.
The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.
NB: Order must match _S_facet_categories definition in locale.cc

collate

```
const category std::locale::collate  [static]
```

Category values.
The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.
NB: Order must match _S_facet_categories definition in locale.cc

ctype

```
const category std::locale::ctype  [static]
```

Category values.
The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.
NB: Order must match _S_facet_categories definition in locale.cc

messages

```
const category std::locale::messages  [static]
```

Category values.
The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.
NB: Order must match _S_facet_categories definition in locale.cc

monetary

```
const category std::locale::monetary  [static]
```

Category values.
The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.
NB: Order must match _S_facet_categories definition in locale.cc

none

```
const category std::locale::none  [static]
```

Category values.
The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.
NB: Order must match _S_facet_categories definition in locale.cc

numeric

```
const category std::locale::numeric  [static]
```

Category values.
The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.

NB: Order must match _S_facet_categories definition in locale.cc

time

```
const category std::locale::time  [static]
```

Category values.
The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.
NB: Order must match _S_facet_categories definition in locale.cc
The documentation for this class was generated from the following files:

- locale_classes.h
- locale_classes.tcc

5.635 std::lock_guard< _Mutex > Class Template Reference

```
#include <mutex>
```

Public Types

- typedef _Mutex **mutex_type**

Public Member Functions

- **lock_guard** (const lock_guard &)=delete
- **lock_guard** (mutex_type &__m)
- **lock_guard** (mutex_type &__m, adopt_lock_t) noexcept
- lock_guard & **operator=** (const lock_guard &)=delete

5.635.1 Detailed Description

template<typename _Mutex>
class std::lock_guard< _Mutex >

A simple scoped lock type.
A lock_guard controls mutex ownership within a scope, releasing ownership in the destructor.

Since

C++11

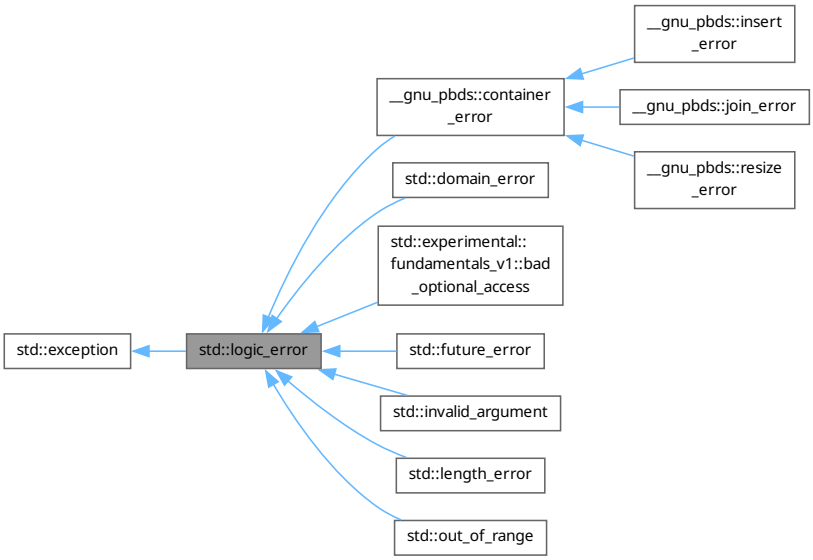
The documentation for this class was generated from the following file:

- std_mutex.h

5.636 std::logic_error Class Reference

```
#include <stdexcept>
```

Inheritance diagram for std::logic_error:



Public Member Functions

- **logic_error** (const char *)
- **logic_error** (const logic_error &) noexcept
- logic_error (const string &__arg)
- **logic_error** (logic_error &&) noexcept
- logic_error & **operator=** (const logic_error &) noexcept
- logic_error & **operator=** (logic_error &&) noexcept
- virtual const char * what () const noexcept

5.636.1 Detailed Description

One of two subclasses of exception.
Logic errors represent problems in the internal logic of a program; in theory, these are preventable, and even detectable before the program runs (e.g., violations of class invariants).

5.636.2 Constructor & Destructor Documentation

logic_error()

```
std::logic_error::logic_error (
            const string & __arg)  [explicit]
```

Takes a character string describing the error.

5.636.3 Member Function Documentation

what()

```
virtual const char * std::logic_error::what () const  [virtual], [noexcept]
```

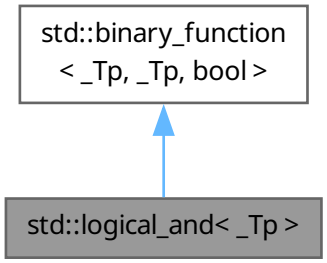
Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from std::exception.
Reimplemented in std::future_error.
The documentation for this class was generated from the following file:

- stdexcept

5.637 std::logical_and< _Tp > Struct Template Reference

`#include <stl_function.h>`
Inheritance diagram for std::logical_and< _Tp >:



Public Types

- typedef _Tp first_argument_type
- typedef bool result_type
- typedef _Tp second_argument_type

Public Member Functions

- constexpr bool **operator()** (const _Tp &__x, const _Tp &__y) const

5.637.1 Detailed Description

template<typename _Tp>
struct std::logical_and< _Tp >

One of the Boolean operations functors.

5.637.2 Member Typedef Documentation

first_argument_type

`typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type [inherited]`
`first_argument_type` is the type of the first argument

result_type

`typedef bool std::binary_function< _Tp, _Tp, bool >::result_type [inherited]`
`result_type` is the return type

second_argument_type

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type  [inherited]
```

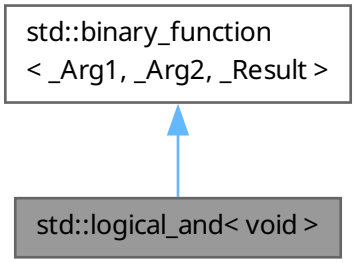
second_argument_type is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

5.638 std::logical_and< void > Struct Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::logical_and< void >:



Public Types

- typedef void first_argument_type
- typedef __is_transparent **is_transparent**
- typedef bool result_type
- typedef void second_argument_type

Public Member Functions

- template<typename _Tp, typename _Up>
  constexpr auto **operator()** (_Tp &&__t, _Up &&__u) const noexcept(noexcept(std::forward< _Tp >(__t) &&std::forward< _Up >(__u))) -> decltype(std::forward< _Tp >(__t) &&std::forward< _Up >(__u))
- constexpr bool **operator()** (const void &__x, const void &__y) const

5.638.1 Detailed Description

One of the Boolean operations functors.

5.638.2 Member Typedef Documentation

first_argument_type

```
typedef void std::binary_function< void, void, bool >::first_argument_type
```

first_argument_type is the type of the first argument

result_type

`typedef bool std::binary_function< void, void, bool >::result_type`
`result_type` is the return type

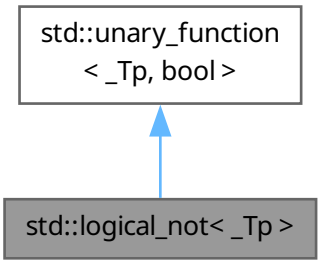
second_argument_type

`typedef void std::binary_function< void, void, bool >::second_argument_type`
`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

5.639 std::logical_not< _Tp > Struct Template Reference

`#include <stl_function.h>`
Inheritance diagram for std::logical_not< _Tp >:



Public Types

- typedef _Tp argument_type
- typedef bool result_type

Public Member Functions

- constexpr bool **operator()** (const _Tp &__x) const

5.639.1 Detailed Description

template<typename _Tp>
struct std::logical_not< _Tp >

One of the Boolean operations functors.

5.639.2 Member Typedef Documentation

argument_type

`typedef _Tp std::unary_function< _Tp, bool >::argument_type [inherited]`
`argument_type` is the type of the argument

result_type

```
typedef bool std::unary_function< _Tp, bool >::result_type  [inherited]
```

result_type is the return type

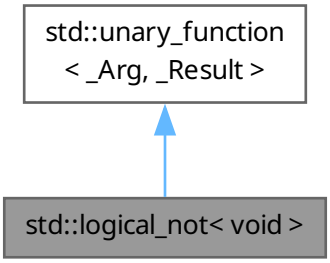
The documentation for this struct was generated from the following file:

- stl_function.h

5.640 std::logical_not< void > Struct Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::logical_not< void >:



Public Types

- typedef void argument_type
- typedef __is_transparent **is_transparent**
- typedef bool result_type

Public Member Functions

- template<typename _Tp>
  constexpr auto **operator()** (_Tp &&__t) const noexcept(noexcept(!std::forward< _Tp >(__t))) -> decltype(!std::forward< _Tp >(__t))
- constexpr bool **operator()** (const void &__x) const

5.640.1 Detailed Description

One of the Boolean operations functors.

5.640.2 Member Typedef Documentation

argument_type

```
typedef void std::unary_function< void, bool >::argument_type
```

argument_type is the type of the argument

result_type

```
typedef bool std::unary_function< void, bool >::result_type
```

result_type is the return type

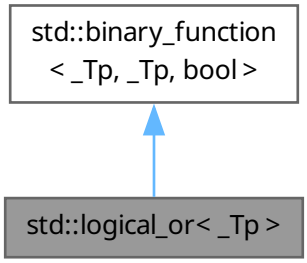
The documentation for this struct was generated from the following file:

- stl_function.h

5.641 std::logical_or< _Tp > Struct Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::logical_or< _Tp >:



Public Types

- typedef _Tp first_argument_type
- typedef bool result_type
- typedef _Tp second_argument_type

Public Member Functions

- constexpr bool **operator()** (const _Tp &__x, const _Tp &__y) const

5.641.1 Detailed Description

template<typename _Tp>
struct std::logical_or< _Tp >

One of the Boolean operations functors.

5.641.2 Member Typedef Documentation

first_argument_type

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type  [inherited]
```

first_argument_type is the type of the first argument

result_type

```
typedef bool std::binary_function< _Tp, _Tp, bool >::result_type  [inherited]
```

result_type is the return type

second_argument_type

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type [inherited]
```

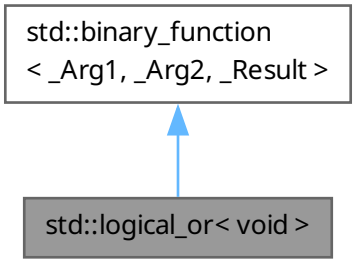
second_argument_type is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

5.642 std::logical_or< void > Struct Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::logical_or< void >:



Public Types

- typedef void first_argument_type
- typedef __is_transparent **is_transparent**
- typedef bool result_type
- typedef void second_argument_type

Public Member Functions

- template<typename _Tp, typename _Up>
  constexpr auto **operator()** (_Tp &&__t, _Up &&__u) const noexcept(noexcept(std::forward< _Tp >(__t)||std::forward< _Up >(__u))) -> decltype(std::forward< _Tp >(__t)||std::forward< _Up >(__u))
- constexpr bool **operator()** (const void &__x, const void &__y) const

5.642.1 Detailed Description

One of the Boolean operations functors.

5.642.2 Member Typedef Documentation

first_argument_type

```
typedef void std::binary_function< void, void, bool >::first_argument_type
```

first_argument_type is the type of the first argument

result_type

`typedef bool std::binary_function< void, void, bool >::result_type`
`result_type` is the return type

second_argument_type

`typedef void std::binary_function< void, void, bool >::second_argument_type`
`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

5.643 std::lognormal_distribution< _RealType > Class Template Reference

`#include <random>`

Classes

- struct param_type

Public Types

- typedef _RealType result_type

Public Member Functions

- **lognormal_distribution** (_RealType __m, _RealType __s=_RealType(1))
- **lognormal_distribution** (const param_type &__p)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- _RealType **m** () const
- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()
- _RealType **s** () const

Friends

- template<typename _RealType1, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::lognormal_distribution< _RealType1 > &__x)
- bool operator== (const lognormal_distribution &__d1, const lognormal_distribution &__d2)

- template<typename _RealType1, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::lognormal_distribution< _RealType1 > &__x)

5.643.1 Detailed Description

template<typename _RealType = double>
class std::lognormal_distribution< _RealType >

A lognormal_distribution random number distribution.
The formula for the normal probability mass function is

$$p(x|m,s) = \frac{1}{sx\sqrt{2\pi}} \exp -\frac{(\ln x - m)^2}{2s^2}$$

Since

C++11

5.643.2 Member Typedef Documentation

result_type

```
template<typename _RealType = double>
typedef _RealType std::lognormal_distribution< _RealType >::result_type
```

The type of the range of the distribution.

5.643.3 Member Function Documentation

max()

```
template<typename _RealType = double>
result_type std::lognormal_distribution< _RealType >::max () const  [inline]
```

Returns the least upper bound value of the distribution.

min()

```
template<typename _RealType = double>
result_type std::lognormal_distribution< _RealType >::min () const  [inline]
```

Returns the greatest lower bound value of the distribution.

operator()()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::lognormal_distribution< _RealType >::operator() (
            _UniformRandomNumberGenerator & __urng)  [inline]
```

Generating functions.
Referenced by operator()().

param() [1/2]

```
template<typename _RealType = double>
param_type std::lognormal_distribution< _RealType >::param () const  [inline]
```

Returns the parameter set of the distribution.

param() [2/2]

```
template<typename _RealType = double>
void std::lognormal_distribution< _RealType >::param (
            const param_type & __param)  [inline]
```

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

reset()

```
template<typename _RealType = double>
void std::lognormal_distribution< _RealType >::reset ()  [inline]
```

Resets the distribution state.

5.643.4 Friends And Related Symbol Documentation

operator<<

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::lognormal_distribution< _RealType1 > & __x)  [friend]
```

Inserts a lognormal_distribution random number distribution __x into the output stream __os.

Parameters

__os	An output stream.
__x	A lognormal_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

operator==

```
template<typename _RealType = double>
bool operator== (
            const lognormal_distribution< _RealType > & __d1,
            const lognormal_distribution< _RealType > & __d2)  [friend]
```

Return true if two lognormal distributions have the same parameters and the sequences that would be generated are equal.

operator>>

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            std::lognormal_distribution< _RealType1 > & __x)  [friend]
```

Extracts a lognormal_distribution random number distribution __x from the input stream __is.

Parameters

_↩ _*is*	An input stream.
_↩ _*x*	A lognormal_distribution random number generator engine.

Returns

The input stream with __x extracted or in an error state.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

5.644 __gnu_pbds::detail::lu_counter_metadata< Size_Type > Class Template Reference

`#include <lu_counter_metadata.hpp>`

Public Types

- typedef Size_Type **size_type**

Friends

- class **lu_counter_policy_base**< **size_type** >

5.644.1 Detailed Description

template<**typename Size_Type = std::size_t**>
class __gnu_pbds::detail::lu_counter_metadata< **Size_Type** >

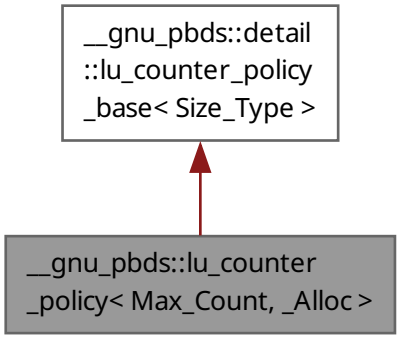
A list-update metadata type that moves elements to the front of the list based on the counter algorithm.
The documentation for this class was generated from the following file:

- lu_counter_metadata.hpp

5.645 __gnu_pbds::lu_counter_policy< Max_Count, _Alloc > Class Template Reference

`#include <list_update_policy.hpp>`

Inheritance diagram for __gnu_pbds::lu_counter_policy< Max_Count, _Alloc >:



Public Types

- enum { max_count }
- typedef _Alloc **allocator_type**
- typedef detail::rebind_traits< _Alloc, metadata_type >::reference metadata_reference
- typedef detail::lu_counter_metadata< size_type > metadata_type
- typedef allocator_type::size_type **size_type**

Public Member Functions

- metadata_type operator() () const
- bool operator() (metadata_reference r_data) const

Private Member Functions

- template<typename Metadata_Reference>
  bool **operator()** (Metadata_Reference r_data, size_type m_max_count) const
- lu_counter_metadata< size_type > **operator()** (size_type max_size) const

5.645.1 Detailed Description

template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>>
class __gnu_pbds::lu_counter_policy< Max_Count, _Alloc >

A list-update policy that moves elements to the front of the list based on the counter algorithm.

5.645.2 Member Typedef Documentation

metadata_reference

```
template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>>
typedef detail::rebind_traits<_Alloc,metadata_type>::reference __gnu_pbds::lu_counter_policy<
Max_Count, _Alloc >::metadata_reference
```

Reference to metadata on which this functor operates.

metadata_type

```
template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>>
typedef detail::lu_counter_metadata<size_type> __gnu_pbds::lu_counter_policy< Max_Count, _Alloc
>::metadata_type
```

Metadata on which this functor operates.

5.645.3 Member Enumeration Documentation

anonymous enum

```
template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>>
anonymous enum
```

Enumerator

max_count	When some element is accessed this number of times, it will be moved to the front of the list.

5.645.4 Member Function Documentation

operator()() [1/2]

```
template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>>
metadata_type __gnu_pbds::lu_counter_policy< Max_Count, _Alloc >::operator() () const  [inline]
```

Creates a metadata object.

References max_count.

operator()() [2/2]

```
template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>>
bool __gnu_pbds::lu_counter_policy< Max_Count, _Alloc >::operator() (
            metadata_reference r_data) const  [inline]
```

Decides whether a metadata object should be moved to the front of the list.

References max_count.

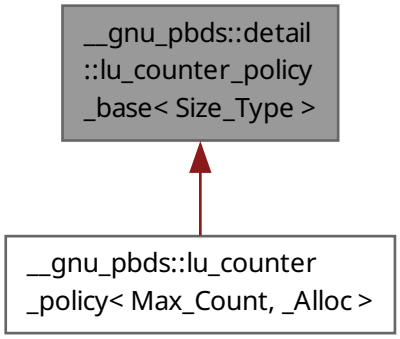
The documentation for this class was generated from the following file:

- list_update_policy.hpp

5.646 __gnu_pbds::detail::lu_counter_policy_base< Size_Type > Class Template Reference

```
#include <lu_counter_metadata.hpp>
```

Inheritance diagram for __gnu_pbds::detail::lu_counter_policy_base< Size_Type >:



Protected Types

- typedef Size_Type **size_type**

Protected Member Functions

- template<typename Metadata_Reference>
  bool **operator()** (Metadata_Reference r_data, size_type m_max_count) const
- lu_counter_metadata< size_type > **operator()** (size_type max_size) const

5.646.1 Detailed Description

template<typename Size_Type>
class __gnu_pbds::detail::lu_counter_policy_base< Size_Type >

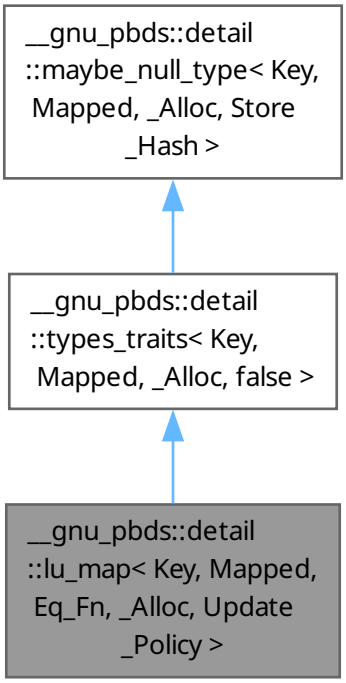
Base class for list-update counter policy.
The documentation for this class was generated from the following file:

- lu_counter_metadata.hpp

5.647 __gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy > Class Template Reference

`#include <lu_map_.hpp>`

Inheritance diagram for __gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >:



Public Types

- typedef _Alloc **allocator_type**
- typedef std::pair< size_type, size_type > **comp_hash**
- typedef const_iterator_ **const_iterator**
- typedef traits_base::const_pointer **const_pointer**
- typedef traits_base::const_reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef Eq_Fn **eq_fn**
- typedef iterator_ **iterator**
- typedef traits_base::key_const_pointer **key_const_pointer**
- typedef traits_base::key_const_reference **key_const_reference**
- typedef traits_base::key_pointer **key_pointer**
- typedef traits_base::key_reference **key_reference**
- typedef traits_base::key_type **key_type**
- typedef traits_base::mapped_const_pointer **mapped_const_pointer**
- typedef traits_base::mapped_const_reference **mapped_const_reference**
- typedef traits_base::mapped_pointer **mapped_pointer**
- typedef traits_base::mapped_reference **mapped_reference**
- typedef traits_base::mapped_type **mapped_type**
- typedef __nothrowcopy::indicator **no_throw_indicator**

- typedef point_const_iterator_ **point_const_iterator**
- typedef point_iterator_ **point_iterator**
- typedef traits_base::pointer **pointer**
- typedef traits_base::reference **reference**
- typedef _Alloc::size_type **size_type**
- typedef integral_constant< int, Store_Hash > **store_extra**
- typedef stored_data< value_type, size_type, Store_Hash > **stored_data_type**
- typedef Update_Policy::metadata_type **update_metadata**
- typedef Update_Policy **update_policy**
- typedef traits_base::value_type **value_type**

Public Member Functions

- **lu_map** (const lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy > &)
- template<typename It>
  lu_map (It, It)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- bool **erase** (key_const_reference)
- template<typename Pred>
  size_type **erase_if** (Pred)
- point_iterator **find** (key_const_reference r_key)
- point_const_iterator **find** (key_const_reference r_key) const
- std::pair< point_iterator, bool > **insert** (const_reference)
- size_type **max_size** () const
- mapped_reference **operator[ ]** (key_const_reference r_key)
- size_type **size** () const
- void **swap** (lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy > &)

Public Attributes

- no_throw_indicator **m_no_throw_copies_indicator**
- no_throw_indicator **m_no_throw_copies_indicator**
- store_extra **m_store_extra_indicator**
- store_extra **m_store_extra_indicator**

Protected Member Functions

- template<typename It>
  void **copy_from_range** (It, It)

Friends

- class **const_iterator_**
- class **iterator_**

5.647.1 Detailed Description

template<typename Key, typename Mapped, typename Eq_Fn, typename _Alloc, typename Update_Policy>
class __gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >

list-based (with updates) associative container. Skip to the lu, my darling.
The documentation for this class was generated from the following file:

- lu_map_.hpp

5.648 __gnu_pbds::lu_move_to_front_policy< _Alloc > Class Template Reference

`#include <list_update_policy.hpp>`

Public Types

- typedef _Alloc **allocator_type**
- typedef detail::rebind_traits< _Alloc, metadata_type >::reference metadata_reference
- typedef null_type metadata_type

Public Member Functions

- metadata_type operator() () const
- bool operator() (metadata_reference r_metadata) const

5.648.1 Detailed Description

template<typename _Alloc = std::allocator<char>>
class __gnu_pbds::lu_move_to_front_policy< _Alloc >

A list-update policy that unconditionally moves elements to the front of the list. A null type means that each link in a list-based container does not actually need metadata.

5.648.2 Member Typedef Documentation

metadata_reference

```
template<typename _Alloc = std::allocator<char>>
typedef detail::rebind_traits<_Alloc,metadata_type>::reference __gnu_pbds::lu_move_to_front_policy<
_Alloc >::metadata_reference
```

Reference to metadata on which this functor operates.

metadata_type

```
template<typename _Alloc = std::allocator<char>>
typedef null_type __gnu_pbds::lu_move_to_front_policy< _Alloc >::metadata_type
```

Metadata on which this functor operates.

5.648.3 Member Function Documentation

operator()() [1/2]

```
template<typename _Alloc = std::allocator<char>>
metadata_type __gnu_pbds::lu_move_to_front_policy< _Alloc >::operator() () const  [inline]
```

Creates a metadata object.

operator()() [2/2]

```
template<typename _Alloc = std::allocator<char>>
bool __gnu_pbds::lu_move_to_front_policy< _Alloc >::operator() (
            metadata_reference r_metadata) const  [inline]
```

Decides whether a metadata object should be moved to the front of the list.
The documentation for this class was generated from the following file:

- list_update_policy.hpp

5.649 std::make_signed< _Tp > Struct Template Reference

`#include <type_traits>`

Public Types

- using **type**

5.649.1 Detailed Description

template<typename _Tp>
struct std::make_signed< _Tp >

make_signed
The documentation for this struct was generated from the following file:

- type_traits

5.650 std::make_unsigned< _Tp > Struct Template Reference

`#include <type_traits>`

Public Types

- using **type**

5.650.1 Detailed Description

template<typename _Tp>
struct std::make_unsigned< _Tp >

make_unsigned
The documentation for this struct was generated from the following file:

- type_traits

5.651 __gnu_cxx::malloc_allocator< _Tp > Class Template Reference

`#include <malloc_allocator.h>`

Public Types

- typedef std::ptrdiff_t **difference_type**
- typedef std::true_type **propagate_on_container_move_assignment**
- typedef std::size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- constexpr **malloc_allocator** (const malloc_allocator &) noexcept
- template<typename _Tp1>
  constexpr **malloc_allocator** (const malloc_allocator< _Tp1 > &) noexcept
- _Tp * **allocate** (size_type __n, const void *=0)
- void **deallocate** (_Tp *__p, size_type)

Friends

- template<typename _Up>
  constexpr bool **operator==** (const malloc_allocator &, const malloc_allocator< _Up > &) noexcept

5.651.1 Detailed Description

template<typename _Tp>
class __gnu_cxx::malloc_allocator< _Tp >

An allocator that uses malloc.
This is precisely the allocator defined in the C++ Standard.

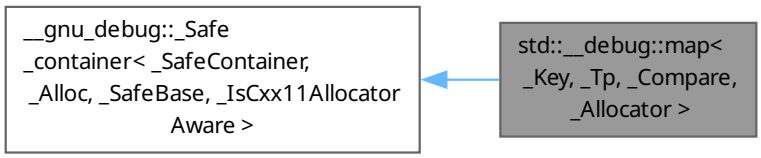
- all allocation calls malloc
- all deallocation calls free

The documentation for this class was generated from the following files:

- experimental/memory_resource
- malloc_allocator.h

5.652 std::__debug::map< _Key, _Tp, _Compare, _Allocator > Class Template Reference

`#include <map.h>`
Inheritance diagram for std::__debug::map< _Key, _Tp, _Compare, _Allocator >:



Public Types

- typedef _Allocator **allocator_type**
- typedef __gnu_debug::_Safe_iterator< _Base_const_iterator, map > **const_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**
- typedef std::reverse_iterator< const_iterator > **const_reverse_iterator**
- typedef _Base::difference_type **difference_type**
- using **insert_return_type**

- typedef __gnu_debug::_Safe_iterator< _Base_iterator, map > **iterator**
- typedef _Compare **key_compare**
- typedef _Key **key_type**
- typedef _Tp **mapped_type**
- using **node_type**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef std::reverse_iterator< iterator > **reverse_iterator**
- typedef _Base::size_type **size_type**
- typedef std::pair< const _Key, _Tp > **value_type**

Public Member Functions

- **map** (_Base_ref __x)
- template<typename _InputIterator>
  map (_InputIterator __first, _InputIterator __last, const _Compare &__comp=_Compare(), const _Allocator &_↩
  _a=_Allocator())
- template<typename _InputIterator>
  map (_InputIterator __first, _InputIterator __last, const allocator_type &__a)
- **map** (const _Compare &__comp, const _Allocator &__a=_Allocator())
- **map** (const allocator_type &__a)
- **map** (const map &)=default
- **map** (const map &__m, const __type_identity_t< allocator_type > &__a)
- **map** (initializer_list< value_type > __l, const _Compare &__c=_Compare(), const allocator_type &__↩
  a=allocator_type())
- **map** (initializer_list< value_type > __l, const allocator_type &__a)
- **map** (map &&)=default
- **map** (map &&__m, const __type_identity_t< allocator_type > &__a) noexcept(noexcept(_Base(std::move(__m), __a)))
- **__attribute** ((__abi_tag__("cxx11"))) iterator erase(iterator __position)
- const _Base & **_M_base** () const noexcept
- _Base & **_M_base** () noexcept
- const_iterator **begin** () const noexcept
- iterator **begin** () noexcept
- const_iterator **cbegin** () const noexcept
- const_iterator **cend** () const noexcept
- void **clear** () noexcept
- const_reverse_iterator **crbegin** () const noexcept
- const_reverse_iterator **crend** () const noexcept
- template<typename... _Args>
  std::pair< iterator, bool > **emplace** (_Args &&... __args)
- template<typename... _Args>
  iterator **emplace_hint** (const_iterator __pos, _Args &&... __args)
- const_iterator **end** () const noexcept
- iterator **end** () noexcept
- template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>
  std::pair< iterator, iterator > **equal_range** (const _Kt &__x)
- template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>
  std::pair< const_iterator, const_iterator > **equal_range** (const _Kt &__x) const
- std::pair< iterator, iterator > **equal_range** (const key_type &__x)
- std::pair< const_iterator, const_iterator > **equal_range** (const key_type &__x) const
- _Base_iterator **erase** (_Base_const_iterator __position)

- size_type **erase** (const key_type &__x)
- iterator **erase** (const_iterator __first, const_iterator __last)
- iterator **erase** (const_iterator __position)
- node_type **extract** (const key_type &__key)
- node_type **extract** (const_iterator __position)
- template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>
  iterator **find** (const _Kt &__x)
- template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>
  const_iterator **find** (const _Kt &__x) const
- iterator **find** (const key_type &__x)
- const_iterator **find** (const key_type &__x) const
- template<typename _InputIterator>
  void **insert** (_InputIterator __first, _InputIterator __last)
- template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
  std::pair< iterator, bool > **insert** (_Pair &&__x)
- std::pair< iterator, bool > **insert** (const value_type &__x)
- iterator **insert** (const_iterator __hint, node_type &&__nh)
- template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
  iterator **insert** (const_iterator __position, _Pair &&__x)
- iterator **insert** (const_iterator __position, const value_type &__x)
- iterator **insert** (const_iterator __position, value_type &&__x)
- insert_return_type **insert** (node_type &&__nh)
- void **insert** (std::initializer_list< value_type > __list)
- std::pair< iterator, bool > **insert** (value_type &&__x)
- template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>
  iterator **lower_bound** (const _Kt &__x)
- template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>
  const_iterator **lower_bound** (const _Kt &__x) const
- iterator **lower_bound** (const key_type &__x)
- const_iterator **lower_bound** (const key_type &__x) const
- map & **operator=** (const map &)=default
- map & **operator=** (initializer_list< value_type > __l)
- map & **operator=** (map &&)=default
- const_reverse_iterator **rbegin** () const noexcept
- reverse_iterator **rbegin** () noexcept
- const_reverse_iterator **rend** () const noexcept
- reverse_iterator **rend** () noexcept
- void **swap** (map &__x) noexcept(/*conditional */)
- template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>
  iterator **upper_bound** (const _Kt &__x)
- template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>
  const_iterator **upper_bound** (const _Kt &__x) const
- iterator **upper_bound** (const key_type &__x)
- const_iterator **upper_bound** (const key_type &__x) const

Protected Member Functions

- constexpr void **_M_swap** (const _Safe_container &__x) const noexcept

Friends

- template<typename _ItT, typename _SeqT, typename _CatT>
  class **::__gnu_debug::_Safe_iterator**

5.652.1 Detailed Description

template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<std::pair<const _Key, _Tp> >>
class std::__debug::map< _Key, _Tp, _Compare, _Allocator >

Class std::map with safety/checking/debug instrumentation.
The documentation for this class was generated from the following file:

- map.h

5.653 std::map< _Key, _Tp, _Compare, _Alloc > Class Template Reference

`#include <map>`

Public Types

- typedef _Alloc **allocator_type**
- typedef _Rep_type::const_iterator **const_iterator**
- typedef _Alloc_traits::const_pointer **const_pointer**
- typedef _Alloc_traits::const_reference **const_reference**
- typedef _Rep_type::const_reverse_iterator **const_reverse_iterator**
- typedef _Rep_type::difference_type **difference_type**
- typedef _Rep_type::iterator **iterator**
- typedef _Compare **key_compare**
- typedef _Key **key_type**
- typedef _Tp **mapped_type**
- typedef _Alloc_traits::pointer **pointer**
- typedef _Alloc_traits::reference **reference**
- typedef _Rep_type::reverse_iterator **reverse_iterator**
- typedef _Rep_type::size_type **size_type**
- typedef std::pair< const _Key, _Tp > **value_type**

Public Member Functions

- map ()=default
- template<typename _InputIterator>
  map (_InputIterator __first, _InputIterator __last)
- template<typename _InputIterator>
  map (_InputIterator __first, _InputIterator __last, const _Compare &__comp, const allocator_type &__↩
  a=allocator_type())
- template<typename _InputIterator>
  map (_InputIterator __first, _InputIterator __last, const allocator_type &__a)
- map (const _Compare &__comp, const allocator_type &__a=allocator_type())
- map (const allocator_type &__a)
- map (const map &)=default
- map (const map &__m, const __type_identity_t< allocator_type > &__a)
- map (initializer_list< value_type > __l, const _Compare &__comp=_Compare(), const allocator_type &__↩
  a=allocator_type())
- map (initializer_list< value_type > __l, const allocator_type &__a)
- map (map &&)=default
- map (map &&__m, const __type_identity_t< allocator_type > &__a) noexcept(is_nothrow_copy_constructible< _Compare >::value &&_Alloc_traits::_S_always_equal())
- ~map ()=default

- mapped_type & at (const key_type &__k)
- const mapped_type & **at** (const key_type &__k) const
- const_iterator begin () const noexcept
- iterator begin () noexcept
- const_iterator cbegin () const noexcept
- const_iterator cend () const noexcept
- void clear () noexcept
- const_reverse_iterator crbegin () const noexcept
- const_reverse_iterator crend () const noexcept
- template<typename... _Args>
  std::pair< iterator, bool > emplace (_Args &&... __args)
- template<typename... _Args>
  iterator emplace_hint (const_iterator __pos, _Args &&... __args)
- bool empty () const noexcept
- const_iterator end () const noexcept
- iterator end () noexcept
- size_type erase (const key_type &__x)
- iterator erase (const_iterator __first, const_iterator __last)
- allocator_type get_allocator () const noexcept
- template<typename _InputIterator>
  void insert (_InputIterator __first, _InputIterator __last)
- void insert (std::initializer_list< value_type > __list)
- template<typename _Obj>
  pair< iterator, bool > insert_or_assign (const key_type &__k, _Obj &&__obj)
- template<typename _Obj>
  iterator insert_or_assign (const_iterator __hint, const key_type &__k, _Obj &&__obj)
- template<typename _Obj>
  iterator **insert_or_assign** (const_iterator __hint, key_type &&__k, _Obj &&__obj)
- template<typename _Obj>
  pair< iterator, bool > **insert_or_assign** (key_type &&__k, _Obj &&__obj)
- key_compare key_comp () const
- size_type max_size () const noexcept
- map & operator= (const map &)=default
- map & operator= (initializer_list< value_type > __l)
- map & operator= (map &&)=default
- mapped_type & operator[ ] (const key_type &__k)
- mapped_type & **operator[ ]** (key_type &&__k)
- const_reverse_iterator rbegin () const noexcept
- reverse_iterator rbegin () noexcept
- const_reverse_iterator rend () const noexcept
- reverse_iterator rend () noexcept
- size_type size () const noexcept
- void swap (map &__x) noexcept(/*conditional */)
- value_compare value_comp () const

- std::pair< iterator, bool > insert (const value_type &__x)
- std::pair< iterator, bool > insert (value_type &&__x)
- template<typename _Pair>
  __enable_if_t< is_constructible< value_type, _Pair >::value, pair< iterator, bool > > insert (_Pair &&__x)

- iterator insert (const_iterator __position, const value_type &__x)
- iterator insert (const_iterator __position, value_type &&__x)
- template<typename _Pair>
  __enable_if_t< is_constructible< value_type, _Pair >::value, iterator > insert (const_iterator __position, _Pair &&__x)

- iterator erase (const_iterator __position)
- __attribute ((__abi_tag__("cxx11"))) iterator erase(iterator __position)

- iterator find (const key_type &__x)
- template<typename _Kt>
  auto find (const _Kt &__x) -> decltype(_M_t._M_find_tr(__x))

- const_iterator find (const key_type &__x) const
- template<typename _Kt>
  auto find (const _Kt &__x) const -> decltype(_M_t._M_find_tr(__x))

- size_type count (const key_type &__x) const
- template<typename _Kt>
  auto count (const _Kt &__x) const -> decltype(_M_t._M_count_tr(__x))

- bool contains (const key_type &__x) const
- template<typename _Kt>
  auto contains (const _Kt &__x) const -> decltype(_M_t._M_find_tr(__x), void(), true)

- iterator lower_bound (const key_type &__x)
- template<typename _Kt>
  auto lower_bound (const _Kt &__x) -> decltype(iterator(_M_t._M_lower_bound_tr(__x)))

- const_iterator lower_bound (const key_type &__x) const
- template<typename _Kt>
  auto lower_bound (const _Kt &__x) const -> decltype(const_iterator(_M_t._M_lower_bound_tr(__x)))

- iterator upper_bound (const key_type &__x)
- template<typename _Kt>
  auto upper_bound (const _Kt &__x) -> decltype(iterator(_M_t._M_upper_bound_tr(__x)))

- const_iterator upper_bound (const key_type &__x) const
- template<typename _Kt>
  auto upper_bound (const _Kt &__x) const -> decltype(const_iterator(_M_t._M_upper_bound_tr(__x)))

- std::pair< iterator, iterator > equal_range (const key_type &__x)
- template<typename _Kt>
  auto equal_range (const _Kt &__x) -> decltype(pair< iterator, iterator >(_M_t._M_equal_range_tr(__x)))

- std::pair< const_iterator, const_iterator > equal_range (const key_type &__x) const
- template<typename _Kt>
  auto equal_range (const _Kt &__x) const -> decltype(pair< const_iterator, const_iterator >(_M_t._M_equal_↩
  range_tr(__x)))

Friends

- template<typename _K1, typename _T1, typename _C1, typename _A1>
  __detail::__synth3way_t< pair< const _K1, _T1 > > **operator<=>** (const map< _K1, _T1, _C1, _A1 > &, const map< _K1, _T1, _C1, _A1 > &)
- template<typename _K1, typename _T1, typename _C1, typename _A1>
  bool **operator==** (const map< _K1, _T1, _C1, _A1 > &, const map< _K1, _T1, _C1, _A1 > &)

5.653.1 Detailed Description

**template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std↩
::allocator<std::pair<const _Key, _Tp> >>**
class std::map< _Key, _Tp, _Compare, _Alloc >

A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.

Since

C++98

Template Parameters

_Key	Type of key objects.
_Tp	Type of mapped objects.
_Compare	Comparison function object type, defaults to less<_Key>.
_Alloc	Allocator type, defaults to allocator<pair<const _Key, _Tp>.

Meets the requirements of a container, a reversible container, and an associative container (using unique keys). For a map<Key,T> the key_type is Key, the mapped_type is T, and the value_type is std::pair<const Key,T>.
Maps support bidirectional iterators.
The private tree data is declared exactly the same way for map and multimap; the distinction is made entirely in how the tree functions are called (*_unique versus *_equal, same as the standard).

5.653.2 Constructor & Destructor Documentation

map() [1/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::map ()  [default]
```

Default constructor creates no elements.

map() [2/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
            const _Compare & __comp,
            const allocator_type & __a = allocator_type())  [inline], [explicit]
```

Creates a map with no elements.

Parameters

__comp	A comparison object.
__a	An allocator object.

map() [3/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
            const map< _Key, _Tp, _Compare, _Alloc > & )  [default]
```

Map copy constructor.
Whether the allocator is copied depends on the allocator traits.

map() [4/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
            map< _Key, _Tp, _Compare, _Alloc > && )  [default]
```

Map move constructor.
The newly-created map contains the exact contents of the moved instance. The moved instance is a valid, but unspecified, map.

map() [5/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
            initializer_list< value_type > __l,
            const _Compare & __comp = _Compare(),
            const allocator_type & __a = allocator_type())  [inline]
```

Builds a map from an initializer_list.

Parameters

__l	An initializer_list.
__comp	A comparison object.
__a	An allocator object.

Create a map consisting of copies of the elements in the initializer_list *__l*. This is linear in N if the range is already sorted, and NlogN otherwise (where N is *__l.size()*).

map() [6/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
            const allocator_type & __a)  [inline], [explicit]
```

Allocator-extended default constructor.

map() [7/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
            const map< _Key, _Tp, _Compare, _Alloc > & __m,
            const __type_identity_t< allocator_type > & __a)  [inline]
```

Allocator-extended copy constructor.

map() [8/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
            map< _Key, _Tp, _Compare, _Alloc > && __m,
            const __type_identity_t< allocator_type > & __a)  [inline], [noexcept]
```

Allocator-extended move constructor.

map() [9/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
            initializer_list< value_type > __l,
            const allocator_type & __a)  [inline]
```

Allocator-extended initialier-list constructor.

map() [10/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _InputIterator>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
            _InputIterator __first,
            _InputIterator __last,
            const allocator_type & __a)  [inline]
```

Allocator-extended range constructor.

map() [11/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _InputIterator>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
            _InputIterator __first,
            _InputIterator __last)  [inline]
```

Builds a map from a range.

Parameters

__first	An input iterator.
__last	An input iterator.

Create a map consisting of copies of the elements from [__first,__last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(__first,__last)).

map() [12/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _InputIterator>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
             _InputIterator __first,
             _InputIterator __last,
             const _Compare & __comp,
             const allocator_type & __a = allocator_type())  [inline]
```

Builds a map from a range.

Parameters

__first	An input iterator.
__last	An input iterator.
__comp	A comparison functor.
__a	An allocator object.

Create a map consisting of copies of the elements from [__first,__last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(__first,__last)).

~map()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::~map ()  [default]
```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

5.653.3 Member Function Documentation

__attribute()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::__attribute (
             (__abi_tag__("cxx11")) )  [inline]
```

Erases an element from a map.

Parameters

__position	An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

at()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
mapped_type & std::map< _Key, _Tp, _Compare, _Alloc >::at (
            const key_type & __k)  [inline]
```

Access to map data.

Parameters

__k	The key for which data should be retrieved.

Returns

A reference to the data whose key is equivalent to *__k*, if such a data is present in the map.

Exceptions

std::out_of_range	If no such data is present.

begin() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::map< _Key, _Tp, _Compare, _Alloc >::begin () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first pair in the map. Iteration is done in ascending order according to the keys.

begin() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map< _Key, _Tp, _Compare, _Alloc >::begin ()  [inline], [noexcept]
```

Returns a read/write iterator that points to the first pair in the map. Iteration is done in ascending order according to the keys.

cbegin()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::map< _Key, _Tp, _Compare, _Alloc >::cbegin () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first pair in the map. Iteration is done in ascending order according to the keys.

cend()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::map< _Key, _Tp, _Compare, _Alloc >::cend () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last pair in the map. Iteration is done in ascending order according to the keys.

clear()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
void std::map< _Key, _Tp, _Compare, _Alloc >::clear ()  [inline], [noexcept]
```

Erases all elements in a map. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

contains() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt>
auto std::map< _Key, _Tp, _Compare, _Alloc >::contains (
            const _Kt & __x) const -> decltype(_M_t._M_find_tr(__x), void(), true)   [inline]
```

Finds whether an element with the given key exists.

Parameters

_↩ _*x*	Key of (key, value) pairs to be located.

Returns

True if there is an element with the specified key.

contains() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
bool std::map< _Key, _Tp, _Compare, _Alloc >::contains (
            const key_type & __x) const  [inline]
```

Finds whether an element with the given key exists.

Parameters

_↩ _*x*	Key of (key, value) pairs to be located.

Returns

True if there is an element with the specified key.

count() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt>
auto std::map< _Key, _Tp, _Compare, _Alloc >::count (
            const _Kt & __x) const -> decltype(_M_t._M_count_tr(__x))   [inline]
```

Finds the number of elements with given key.

Parameters

_↩ _x	Key of (key, value) pairs to be located.

Returns

Number of elements with specified key.

This function only makes sense for multimaps; for map the result will either be 0 (not present) or 1 (present).

count() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::map< _Key, _Tp, _Compare, _Alloc >::count (
            const key_type & __x) const  [inline]
```

Finds the number of elements with given key.

Parameters

_↩ _x	Key of (key, value) pairs to be located.

Returns

Number of elements with specified key.

This function only makes sense for multimaps; for map the result will either be 0 (not present) or 1 (present).

crbegin()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::crbegin () const  [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the map. Iteration is done in descending order according to the keys.

crend()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::crend () const  [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first pair in the map. Iteration is done in descending order according to the keys.

emplace()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename... _Args>
std::pair< iterator, bool > std::map< _Key, _Tp, _Compare, _Alloc >::emplace (
            _Args &&... __args)  [inline]
```

Attempts to build and insert a std::pair into the map.

Parameters

__args	Arguments used to generate a new pair instance (see std::piecewise_contruct for passing arguments to each part of the pair constructor).

Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to build and insert a (key, value) pair into the map. A map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the map.
Insertion requires logarithmic time.

emplace_hint()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename... _Args>
iterator std::map< _Key, _Tp, _Compare, _Alloc >::emplace_hint (
            const_iterator __pos,
            _Args &&... __args)  [inline]
```

Attempts to build and insert a std::pair into the map.

Parameters

__pos	An iterator that serves as a hint as to where the pair should be inserted.
__args	Arguments used to generate a new pair instance (see std::piecewise_contruct for passing arguments to each part of the pair constructor).

Returns

An iterator that points to the element with key of the std::pair built from *__args* (may or may not be that std::pair).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument emplace() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
See https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints for more on *hinting*.
Insertion requires logarithmic time (if the hint is not taken).
Referenced by std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::emplace(), std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert(), std::map< _Key, _Tp, _Cmp, polymorphic_ and std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert_or_assign().

empty()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
bool std::map< _Key, _Tp, _Compare, _Alloc >::empty () const  [inline], [nodiscard], [noexcept]
```

Returns true if the map is empty. (Thus begin() would equal end().)

end() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::map< _Key, _Tp, _Compare, _Alloc >::end () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last pair in the map. Iteration is done in ascending order according to the keys.

end() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map< _Key, _Tp, _Compare, _Alloc >::end ()  [inline], [noexcept]
```

Returns a read/write iterator that points one past the last pair in the map. Iteration is done in ascending order according to the keys.

Referenced by std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::at(), std::map< _Key, _Tp, _Cmp, po
std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert(), std::map< _Key, _Tp, _Cmp, polymorphic_
and std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::operator[]().

equal_range() [1/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt>
auto std::map< _Key, _Tp, _Compare, _Alloc >::equal_range (
            const _Kt & __x) -> decltype(pair<iterator, iterator>(_M_t._M_equal_range_tr(__x)))
[inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key of (key, value) pairs to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
               c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multimaps.

equal_range() [2/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt>
```

```
auto std::map< _Key, _Tp, _Compare, _Alloc >::equal_range (
            const _Kt & __x) const -> decltype(pair<const_iterator, const_iterator>( _M_t._M_↩
equal_range_tr(__x)))   [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key of (key, value) pairs to be located.

Returns

Pair of read-only (constant) iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
            c.upper_bound(val))
```

(but is faster than making the calls separately).
This function probably only makes sense for multimaps.

equal_range() [3/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::pair< iterator, iterator > std::map< _Key, _Tp, _Compare, _Alloc >::equal_range (
            const key_type & __x)  [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key of (key, value) pairs to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
            c.upper_bound(val))
```

(but is faster than making the calls separately).
This function probably only makes sense for multimaps.

equal_range() [4/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::pair< const_iterator, const_iterator > std::map< _Key, _Tp, _Compare, _Alloc >::equal_range
(
            const key_type & __x) const  [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key of (key, value) pairs to be located.

Returns

Pair of read-only (constant) iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
               c.upper_bound(val))
```

(but is faster than making the calls separately).
This function probably only makes sense for multimaps.

erase() [1/3]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::map< _Key, _Tp, _Compare, _Alloc >::erase (
             const key_type & __x)  [inline]
```

Erases elements according to the provided key.

Parameters

_↩ _x	Key of element to be erased.

Returns

The number of elements erased.

This function erases all the elements located by the given key from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [2/3]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map< _Key, _Tp, _Compare, _Alloc >::erase (
             const_iterator __first,
             const_iterator __last)  [inline]
```

Erases a [first,last) range of elements from a map.

Parameters

__first	Iterator pointing to the start of the range to be erased.
__last	Iterator pointing to the end of the range to be erased.

Returns

The iterator *__last*.

This function erases a sequence of elements from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [3/3]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map< _Key, _Tp, _Compare, _Alloc >::erase (
            const_iterator __position)  [inline]
```

Erases an element from a map.

Parameters

__position	An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

find() [1/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt>
auto std::map< _Key, _Tp, _Compare, _Alloc >::find (
            const _Kt & __x) -> decltype(_M_t._M_find_tr(__x))   [inline]
```

Tries to locate an element in a map.

Parameters

_↩ *_x*	Key of (key, value) pair to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end (`end()`) iterator.

find() [2/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt>
auto std::map< _Key, _Tp, _Compare, _Alloc >::find (
            const _Kt & __x) const -> decltype(_M_t._M_find_tr(__x))   [inline]
```

Tries to locate an element in a map.

Parameters

_↩ *_x*	Key of (key, value) pair to be located.

Returns

Read-only (constant) iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns a constant iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end (`end()`) iterator.

find() [3/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map< _Key, _Tp, _Compare, _Alloc >::find (
            const key_type & __x)  [inline]
```

Tries to locate an element in a map.

Parameters

_↩ _x	Key of (key, value) pair to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end (`end()`) iterator.

find() [4/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::map< _Key, _Tp, _Compare, _Alloc >::find (
            const key_type & __x) const  [inline]
```

Tries to locate an element in a map.

Parameters

_↩ _x	Key of (key, value) pair to be located.

Returns

Read-only (constant) iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns a constant iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end (`end()`) iterator.

get_allocator()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
allocator_type std::map< _Key, _Tp, _Compare, _Alloc >::get_allocator () const  [inline], [noexcept]
```

Get a copy of the memory allocation object.

insert() [1/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _InputIterator>
void std::map< _Key, _Tp, _Compare, _Alloc >::insert (
            _InputIterator __first,
            _InputIterator __last)  [inline]
```

Template function that attempts to insert a range of elements.

Parameters

__first	Iterator pointing to the start of the range to be inserted.
__last	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

insert() [2/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Pair>
__enable_if_t< is_constructible< value_type, _Pair >::value, pair< iterator, bool > > std::map<
_Key, _Tp, _Compare, _Alloc >::insert (
            _Pair && __x)  [inline]
```

Attempts to insert a std::pair into the map.

Parameters

_↩ _x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the map. A map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the map.
Insertion requires logarithmic time.

insert() [3/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::pair< iterator, bool > std::map< _Key, _Tp, _Compare, _Alloc >::insert (
            const value_type & __x)  [inline]
```

Attempts to insert a std::pair into the map.

Parameters

_↩ _x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the map. A map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the map.
Insertion requires logarithmic time.
Referenced by std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert(), and std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::operator[]().

insert() [4/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Pair>
__enable_if_t< is_constructible< value_type, _Pair >::value, iterator > std::map< _Key, _Tp, _↩
Compare, _Alloc >::insert (
            const_iterator __position,
            _Pair && __x)  [inline]
```

Attempts to insert a std::pair into the map.

Parameters

__position	An iterator that serves as a hint as to where the pair should be inserted.
__x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the element with key of *__x* (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument insert() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
See https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.↩associative.insert_hints for more on *hinting*.
Insertion requires logarithmic time (if the hint is not taken).

insert() [5/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map< _Key, _Tp, _Compare, _Alloc >::insert (
            const_iterator __position,
            const value_type & __x)  [inline]
```

Attempts to insert a std::pair into the map.

Parameters

__position	An iterator that serves as a hint as to where the pair should be inserted.
__x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the element with key of __x (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument insert() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
See https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints for more on *hinting*.
Insertion requires logarithmic time (if the hint is not taken).

insert() [6/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map< _Key, _Tp, _Compare, _Alloc >::insert (
            const_iterator __position,
            value_type && __x)  [inline]
```

Attempts to insert a std::pair into the map.

Parameters

__position	An iterator that serves as a hint as to where the pair should be inserted.
__x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the element with key of __x (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument insert() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
See https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints for more on *hinting*.
Insertion requires logarithmic time (if the hint is not taken).

insert() [7/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
void std::map< _Key, _Tp, _Compare, _Alloc >::insert (
            std::initializer_list< value_type > __list)  [inline]
```

Attempts to insert a list of std::pairs into the map.

Parameters

__list	A std::initializer_list<value_type> of pairs to be inserted.

Complexity similar to that of the range constructor.

insert() [8/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::pair< iterator, bool > std::map< _Key, _Tp, _Compare, _Alloc >::insert (
            value_type && __x)  [inline]
```

Attempts to insert a std::pair into the map.

Parameters

_↩ _x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the map. A map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the map.
Insertion requires logarithmic time.

insert_or_assign() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Obj>
pair< iterator, bool > std::map< _Key, _Tp, _Compare, _Alloc >::insert_or_assign (
            const key_type & __k,
            _Obj && __obj)  [inline]
```

Attempts to insert or assign a std::pair into the map.

Parameters

__k	Key to use for finding a possibly existing pair in the map.
__obj	Argument used to generate the .second for a pair instance.

Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the map. A map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the map. If the pair was already in the map, the .second of the pair is assigned from __obj.
Insertion requires logarithmic time.

insert_or_assign() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Obj>
iterator std::map< _Key, _Tp, _Compare, _Alloc >::insert_or_assign (
            const_iterator __hint,
            const key_type & __k,
            _Obj && __obj)  [inline]
```

Attempts to insert or assign a std::pair into the map.

Parameters

__hint	An iterator that serves as a hint as to where the pair should be inserted.
__k	Key to use for finding a possibly existing pair in the map.
__obj	Argument used to generate the .second for a pair instance.

Returns

An iterator that points to the element with key of __x (may or may not be the pair passed in).

This function attempts to insert a (key, value) pair into the map. A map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the map. If the pair was already in the map, the .second of the pair is assigned from __obj.
Insertion requires logarithmic time.

key_comp()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
key_compare std::map< _Key, _Tp, _Compare, _Alloc >::key_comp () const  [inline]
```

Returns the key comparison object out of which the map was constructed.
Referenced by std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::at(), std::map< _Key, _Tp, _Cmp, po
std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert(), std::map< _Key, _Tp, _Cmp, polymorphic_
and std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::operator[]().

lower_bound() [1/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt>
auto std::map< _Key, _Tp, _Compare, _Alloc >::lower_bound (
             const _Kt & __x) -> decltype(iterator(_M_t._M_lower_bound_tr(__x)))   [inline]
```

Finds the beginning of a subsequence matching given key.

Parameters

_↩ _x	Key of (key, value) pair to be located.

Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

lower_bound() [2/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt>
auto std::map< _Key, _Tp, _Compare, _Alloc >::lower_bound (
             const _Kt & __x) const -> decltype(const_iterator(_M_t._M_lower_bound_tr(__x)))
[inline]
```

Finds the beginning of a subsequence matching given key.

Parameters

_↩ _x	Key of (key, value) pair to be located.

Returns

Read-only (constant) iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

lower_bound() [3/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map< _Key, _Tp, _Compare, _Alloc >::lower_bound (
            const key_type & __x)  [inline]
```

Finds the beginning of a subsequence matching given key.

Parameters

_↩ _x	Key of (key, value) pair to be located.

Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.
Referenced by std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::at(), std::map< _Key, _Tp, _Cmp, po
std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert(), std::map< _Key, _Tp, _Cmp, polymorphic_
and std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::operator[]().

lower_bound() [4/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::map< _Key, _Tp, _Compare, _Alloc >::lower_bound (
            const key_type & __x) const  [inline]
```

Finds the beginning of a subsequence matching given key.

Parameters

_↩ _x	Key of (key, value) pair to be located.

Returns

Read-only (constant) iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

max_size()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::map< _Key, _Tp, _Compare, _Alloc >::max_size () const  [inline], [noexcept]
```

Returns the maximum size of the map.

operator=() [1/3]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
map & std::map< _Key, _Tp, _Compare, _Alloc >::operator= (
            const map< _Key, _Tp, _Compare, _Alloc > & )  [default]
```

Map assignment operator.
Whether the allocator is copied depends on the allocator traits.

operator=() [2/3]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
map & std::map< _Key, _Tp, _Compare, _Alloc >::operator= (
            initializer_list< value_type > __l)  [inline]
```

Map list assignment operator.

Parameters

↩ _↩ ↩ _↩ *l*	An initializer_list.

This function fills a map with copies of the elements in the initializer list *__l*.
Note that the assignment completely changes the map and that the resulting map's size is the same as the number of elements assigned.

operator=() [3/3]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
map & std::map< _Key, _Tp, _Compare, _Alloc >::operator= (
            map< _Key, _Tp, _Compare, _Alloc > && )  [default]
```

Move assignment operator.

operator[]()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
mapped_type & std::map< _Key, _Tp, _Compare, _Alloc >::operator[] (
            const key_type & __k)  [inline]
```

Subscript ([]) access to map data.

Parameters

_↩ *_k*	The key for which data should be retrieved.

Returns

A reference to the data of the (key,data) pair.

Allows for easy lookup with the subscript ([]) operator. Returns data associated with the key specified in subscript. If the key does not exist, a pair with that key is created using default values, which is then returned.
Lookup requires logarithmic time.

rbegin() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::rbegin () const  [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the map. Iteration is done in descending order according to the keys.

rbegin() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::rbegin ()  [inline], [noexcept]
```

Returns a read/write reverse iterator that points to the last pair in the map. Iteration is done in descending order according to the keys.

rend() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::rend () const  [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first pair in the map. Iteration is done in descending order according to the keys.

rend() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::rend ()  [inline], [noexcept]
```

Returns a read/write reverse iterator that points to one before the first pair in the map. Iteration is done in descending order according to the keys.

size()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::map< _Key, _Tp, _Compare, _Alloc >::size () const  [inline], [noexcept]
```

Returns the size of the map.

swap()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
void std::map< _Key, _Tp, _Compare, _Alloc >::swap (
            map< _Key, _Tp, _Compare, _Alloc > & __x)  [inline], [noexcept]
```

Swaps data with another map.

Parameters

_↩ _x	A map of the same element and allocator types.

This exchanges the elements between two maps in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global std::swap() function is specialized such that std::swap(m1,m2) will feed to this function.
Whether the allocators are swapped depends on the allocator traits.

upper_bound() [1/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt>
auto std::map< _Key, _Tp, _Compare, _Alloc >::upper_bound (
            const _Kt & __x) -> decltype(iterator(_M_t._M_upper_bound_tr(__x)))   [inline]
```

Finds the end of a subsequence matching given key.

Parameters

_↩ _x	Key of (key, value) pair to be located.

Returns

Iterator pointing to the first element greater than key, or end().

upper_bound() [2/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt>
auto std::map< _Key, _Tp, _Compare, _Alloc >::upper_bound (
            const _Kt & __x) const -> decltype(const_iterator(_M_t._M_upper_bound_tr(__x)))
[inline]
```

Finds the end of a subsequence matching given key.

Parameters

_↩ _x	Key of (key, value) pair to be located.

Returns

Read-only (constant) iterator pointing to first iterator greater than key, or end().

upper_bound() [3/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map< _Key, _Tp, _Compare, _Alloc >::upper_bound (
            const key_type & __x)  [inline]
```

Finds the end of a subsequence matching given key.

Parameters

_↩ _x	Key of (key, value) pair to be located.

Returns

Iterator pointing to the first element greater than key, or end().

upper_bound() [4/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::map< _Key, _Tp, _Compare, _Alloc >::upper_bound (
            const key_type & __x) const  [inline]
```

Finds the end of a subsequence matching given key.

Parameters

_↩ _x	Key of (key, value) pair to be located.

Returns

Read-only (constant) iterator pointing to first iterator greater than key, or end().

value_comp()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
value_compare std::map< _Key, _Tp, _Compare, _Alloc >::value_comp () const  [inline]
```

Returns a value comparison object, built from the key comparison object out of which the map was constructed.
The documentation for this class was generated from the following file:

- stl_map.h

5.654 std::mask_array< _Tp > Class Template Reference

`#include <mask_array.h>`

Public Types

- typedef _Tp **value_type**

Public Member Functions

- mask_array (const mask_array &)
- template<class _Dom>
  void **operator%=** (const _Expr< _Dom, _Tp > &) const
- void operator%= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator&=** (const _Expr< _Dom, _Tp > &) const
- void operator&= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator*=** (const _Expr< _Dom, _Tp > &) const
- void operator*= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator+=** (const _Expr< _Dom, _Tp > &) const
- void operator+= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator-=** (const _Expr< _Dom, _Tp > &) const
- void operator-= (const valarray< _Tp > &) const

- template<class _Dom>
  void **operator/=** (const _Expr< _Dom, _Tp > &) const
- void operator/= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator<<=** (const _Expr< _Dom, _Tp > &) const
- void operator<<= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator=** (const _Expr< _Dom, _Tp > &) const
- template<class _Ex>
  void **operator=** (const _Expr< _Ex, _Tp > &__e) const
- void operator= (const _Tp &) const
- mask_array & operator= (const mask_array &)
- void **operator=** (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator>>=** (const _Expr< _Dom, _Tp > &) const
- void operator>>= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator^=** (const _Expr< _Dom, _Tp > &) const
- void operator^= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator|=** (const _Expr< _Dom, _Tp > &) const
- void operator|= (const valarray< _Tp > &) const

Friends

- class **valarray< _Tp >**

5.654.1 Detailed Description

template<class _Tp>
class std::mask_array< _Tp >

Reference to selected subset of an array.
A mask_array is a reference to the actual elements of an array specified by a bitmask in the form of an array of bool. The way to get a mask_array is to call operator[](valarray<bool>) on a valarray. The returned mask_array then permits carrying operations out on the referenced subset of elements in the original valarray.
For example, if a mask_array is obtained using the array (false, true, false, true) as an argument, the mask array has two elements referring to array[1] and array[3] in the underlying array.

Parameters

Tp	Element type.

5.654.2 Member Function Documentation

operator%=()

```
template<class _Tp>
void std::mask_array< _Tp >::operator%= (
            const valarray< _Tp > & ) const
```

Modulo slice elements by corresponding elements of *v*.

operator&=()

```
template<class _Tp>
void std::mask_array< _Tp >::operator&= (
            const valarray< _Tp > & ) const
```

Logical and slice elements with corresponding elements of *v*.

operator*=()

```
template<class _Tp>
void std::mask_array< _Tp >::operator*= (
            const valarray< _Tp > & ) const
```

Multiply slice elements by corresponding elements of *v*.

operator+=()

```
template<class _Tp>
void std::mask_array< _Tp >::operator+= (
            const valarray< _Tp > & ) const
```

Add corresponding elements of *v* to slice elements.

operator-=()

```
template<class _Tp>
void std::mask_array< _Tp >::operator-= (
            const valarray< _Tp > & ) const
```

Subtract corresponding elements of *v* from slice elements.

operator/=()

```
template<class _Tp>
void std::mask_array< _Tp >::operator/= (
            const valarray< _Tp > & ) const
```

Divide slice elements by corresponding elements of *v*.

operator<<=()

```
template<class _Tp>
void std::mask_array< _Tp >::operator<<= (
            const valarray< _Tp > & ) const
```

Left shift slice elements by corresponding elements of *v*.

operator>>=()

```
template<class _Tp>
void std::mask_array< _Tp >::operator>>= (
            const valarray< _Tp > & ) const
```

Right shift slice elements by corresponding elements of *v*.
References mask_array().

operator^=()

```
template<class _Tp>
void std::mask_array< _Tp >::operator^= (
            const valarray< _Tp > & ) const
```

Logical xor slice elements with corresponding elements of *v*.

operator"" |=()

```
template<class _Tp>
void std::mask_array< _Tp >::operator|= (
            const valarray< _Tp > & ) const
```

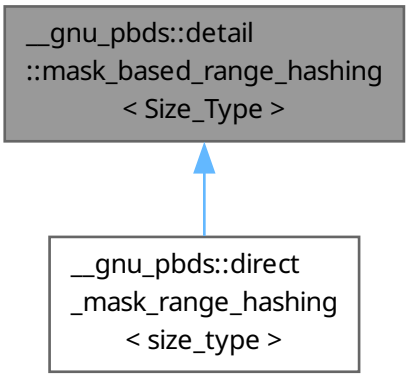
Logical or slice elements with corresponding elements of *v*.
The documentation for this class was generated from the following files:

- valarray
- mask_array.h

5.655 __gnu_pbds::detail::mask_based_range_hashing< Size_Type > Class Template Reference

```
#include <mask_based_range_hashing.hpp>
```

Inheritance diagram for __gnu_pbds::detail::mask_based_range_hashing< Size_Type >:



Protected Types

- typedef Size_Type **size_type**

Protected Member Functions

- void **notify_resized** (size_type size)
- size_type **range_hash** (size_type hash) const
- void **swap** (mask_based_range_hashing &other)

5.655.1 Detailed Description

template<typename Size_Type>
class __gnu_pbds::detail::mask_based_range_hashing< Size_Type >

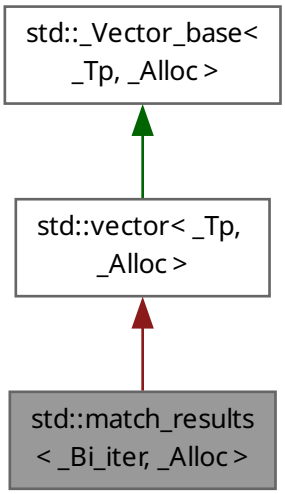
Range hashing policy.
The documentation for this class was generated from the following file:

- mask_based_range_hashing.hpp

5.656 std::match_results< _Bi_iter, _Alloc > Class Template Reference

`#include <regex>`

Inheritance diagram for std::match_results< _Bi_iter, _Alloc >:



Public Types

28.10 Public Types

- typedef sub_match< _Bi_iter > **value_type**
- typedef const value_type & **const_reference**
- typedef value_type & **reference**
- typedef _Base_type::const_iterator **const_iterator**
- typedef const_iterator **iterator**
- typedef __iter_traits::difference_type **difference_type**
- typedef allocator_traits< _Alloc >::size_type **size_type**
- typedef _Alloc **allocator_type**
- typedef __iter_traits::value_type **char_type**
- typedef std::basic_string< char_type > **string_type**

Public Member Functions

- template<typename _Out_iter>
  _Out_iter **format** (_Out_iter __out, const match_results< _Bi_iter, _Alloc >::char_type *__fmt_first, const match_results< _Bi_iter, _Alloc >::char_type *__fmt_last, match_flag_type __flags) const
- bool ready () const noexcept

28.10.1 Construction, Copying, and Destruction

- match_results ()
- match_results (const _Alloc &__a) noexcept
- match_results (const match_results &)=default
- match_results (match_results &&) noexcept=default

- match_results & operator= (const match_results &)=default
- match_results & operator= (match_results &&)=default
- ~match_results ()=default
- match_results (const match_results &__m, const _Alloc &__a)
- match_results (match_results &&__m, const _Alloc &__a) noexcept(noexcept(_Base_type(std::move(__m), ↩ __a)))

28.10.2 Size

- size_type size () const noexcept
- size_type max_size () const noexcept
- bool empty () const noexcept

28.10.4 Element Access

- difference_type length (size_type __sub=0) const
- difference_type position (size_type __sub=0) const
- string_type str (size_type __sub=0) const
- const_reference operator[ ] (size_type __sub) const
- const_reference prefix () const
- const_reference suffix () const
- const_iterator begin () const noexcept
- const_iterator cbegin () const noexcept
- const_iterator end () const noexcept
- const_iterator cend () const noexcept

28.10.5 Formatting

These functions perform formatted substitution of the matched character sequences into their target. The format specifiers and escape sequences accepted by these functions are determined by their `flags` parameter as documented above.

- template<typename _Out_iter>
  _Out_iter format (_Out_iter __out, const char_type *__fmt_first, const char_type *__fmt_last, match_flag_type __flags=regex_constants::format_default) const
- template<typename _Out_iter, typename _St, typename _Sa>
  _Out_iter format (_Out_iter __out, const basic_string< char_type, _St, _Sa > &__fmt, match_flag_type __↩ flags=regex_constants::format_default) const
- template<typename _St, typename _Sa>
  basic_string< char_type, _St, _Sa > format (const basic_string< char_type, _St, _Sa > &__fmt, match_flag_type __flags=regex_constants::format_default) const
- string_type format (const char_type *__fmt, match_flag_type __flags=regex_constants::format_default) const

28.10.6 Allocator

- allocator_type get_allocator () const noexcept

28.10.7 Swap

- void swap (match_results &__that) noexcept

Private Member Functions

- constexpr iterator begin () noexcept
- constexpr iterator end () noexcept
- constexpr const_reference operator[ ] (size_type __n) const noexcept
- constexpr reference operator[ ] (size_type __n) noexcept
- constexpr void swap (vector &__x) noexcept

Friends

- template<typename, typename, typename>
  class **regex_iterator**

Related Symbols

(Note that these are not member symbols.)

- template<typename _Bi_iter, typename _Alloc>
  bool operator== (const match_results< _Bi_iter, _Alloc > &__m1, const match_results< _Bi_iter, _Alloc > &_↩
  _m2)
- template<typename _Bi_iter, typename _Alloc>
  void swap (match_results< _Bi_iter, _Alloc > &__lhs, match_results< _Bi_iter, _Alloc > &__rhs) noexcept

5.656.1 Detailed Description

template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
class std::match_results< _Bi_iter, _Alloc >

The results of a match or search operation.
A collection of character sequences representing the result of a regular expression match. Storage for the collection is allocated and freed as necessary by the member functions of class template match_results.
This class satisfies the Sequence requirements, with the exception that only the operations defined for a const-qualified Sequence are supported.
The sub_match object stored at index 0 represents sub-expression 0, i.e. the whole match. In this case the sub_↩
match member matched is always true. The sub_match object stored at index n denotes what matched the marked sub-expression n within the matched expression. If the sub-expression n participated in a regular expression match then the sub_match member matched evaluates to true, and members first and second denote the range of characters [first, second) which formed that match. Otherwise matched is false, and members first and second point to the end of the sequence that was searched.

Since

C++11

5.656.2 Constructor & Destructor Documentation

match_results() [1/6]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
std::match_results< _Bi_iter, _Alloc >::match_results ()  [inline]
```

Constructs a default match_results container.

Postcondition

size() returns 0 and str() returns an empty string.

match_results() [2/6]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
std::match_results< _Bi_iter, _Alloc >::match_results (
            const _Alloc & __a)  [inline], [explicit], [noexcept]
```

Constructs a default match_results container.

Postcondition

size() returns 0 and str() returns an empty string.

match_results() [3/6]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
std::match_results< _Bi_iter, _Alloc >::match_results (
            const match_results< _Bi_iter, _Alloc > & ) [default]
```

Copy constructs a match_results.

match_results() [4/6]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
std::match_results< _Bi_iter, _Alloc >::match_results (
            match_results< _Bi_iter, _Alloc > && ) [default], [noexcept]
```

Move constructs a match_results.

~match_results()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
std::match_results< _Bi_iter, _Alloc >::~match_results () [default]
```

Destroys a match_results object.

match_results() [5/6]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
std::match_results< _Bi_iter, _Alloc >::match_results (
            const match_results< _Bi_iter, _Alloc > & __m,
            const _Alloc & __a) [inline]
```

Constructs a default match_results container.

Postcondition

size() returns 0 and str() returns an empty string.

match_results() [6/6]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
std::match_results< _Bi_iter, _Alloc >::match_results (
            match_results< _Bi_iter, _Alloc > && __m,
            const _Alloc & __a) [inline], [noexcept]
```

Constructs a default match_results container.

Postcondition

size() returns 0 and str() returns an empty string.

5.656.3 Member Function Documentation

begin()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
const_iterator std::match_results< _Bi_iter, _Alloc >::begin () const [inline], [noexcept]
```

Gets an iterator to the start of the sub_match collection.

Referenced by std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const cha
and std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const char * >::ope

cbegin()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
const_iterator std::match_results< _Bi_iter, _Alloc >::cbegin () const  [inline], [noexcept]
```

Gets an iterator to the start of the sub_match collection.

cend()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
const_iterator std::match_results< _Bi_iter, _Alloc >::cend () const  [inline], [noexcept]
```

Gets an iterator to one-past-the-end of the collection.

empty()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
bool std::match_results< _Bi_iter, _Alloc >::empty () const  [inline], [nodiscard], [noexcept]
```

Indicates if the match_results contains no results.

Return values

true	The match_results object is empty.
false	The match_results object is not empty.

Referenced by std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const cha
std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const char * >::prefix(),
and std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const char * >::suffi

end()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
const_iterator std::match_results< _Bi_iter, _Alloc >::end () const  [inline], [noexcept]
```

Gets an iterator to one-past-the-end of the collection.
Referenced by std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const cha
and std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const char * >::oper

format() [1/4]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
template<typename _Out_iter, typename _St, typename _Sa>
_Out_iter std::match_results< _Bi_iter, _Alloc >::format (
            _Out_iter __out,
            const basic_string< char_type, _St, _Sa > & __fmt,
            match_flag_type __flags = regex_constants::format_default) const  [inline]
```

Precondition

ready() == true

format() [2/4]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
template<typename _Out_iter>
_Out_iter std::match_results< _Bi_iter, _Alloc >::format (
            _Out_iter __out,
            const char_type * __fmt_first,
            const char_type * __fmt_last,
            match_flag_type __flags = regex_constants::format_default) const
```

Precondition

ready() == true

Referenced by std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const cha
std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const char * >::format(),
and std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const char * >::form

format() [3/4]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
template<typename _St, typename _Sa>
basic_string< char_type, _St, _Sa > std::match_results< _Bi_iter, _Alloc >::format (
            const basic_string< char_type, _St, _Sa > & __fmt,
            match_flag_type __flags = regex_constants::format_default) const  [inline]
```

Precondition

ready() == true

format() [4/4]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
string_type std::match_results< _Bi_iter, _Alloc >::format (
            const char_type * __fmt,
            match_flag_type __flags = regex_constants::format_default) const  [inline]
```

Precondition

ready() == true

get_allocator()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
allocator_type std::match_results< _Bi_iter, _Alloc >::get_allocator () const  [inline], [noexcept]
```

Gets a copy of the allocator.

length()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
difference_type std::match_results< _Bi_iter, _Alloc >::length (
            size_type __sub = 0) const  [inline]
```

Gets the length of the indicated submatch.

Parameters

__sub	indicates the submatch.

Precondition

ready() == true

This function returns the length of the indicated submatch, or the length of the entire match if __sub is zero (the default).

max_size()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
size_type std::match_results< _Bi_iter, _Alloc >::max_size () const  [inline], [noexcept]
```

Gets the number of matches and submatches.
The number of matches for a given regular expression will be either 0 if there was no match or mark_count() + 1 if a match was successful. Some matches may be empty.

Returns

the number of matches found.

operator=() [1/2]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
match_results & std::match_results< _Bi_iter, _Alloc >::operator= (
            const match_results< _Bi_iter, _Alloc > & )  [default]
```

Assigns rhs to *this.

operator=() [2/2]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
match_results & std::match_results< _Bi_iter, _Alloc >::operator= (
            match_results< _Bi_iter, _Alloc > && )  [default]
```

Move-assigns rhs to *this.

operator[]()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
const_reference std::match_results< _Bi_iter, _Alloc >::operator[] (
            size_type __sub) const  [inline]
```

Gets a sub_match reference for the match or submatch.

Parameters

__sub	indicates the submatch.

Precondition

ready() == true

This function gets a reference to the indicated submatch, or the entire match if __sub is zero.
If __sub >= size() then this function returns a sub_match with a special value indicating no submatch.

position()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
difference_type std::match_results< _Bi_iter, _Alloc >::position (
            size_type __sub = 0) const  [inline]
```

Gets the offset of the beginning of the indicated submatch.

Parameters

__sub	indicates the submatch.

Precondition

ready() == true

This function returns the offset from the beginning of the target sequence to the beginning of the submatch, unless the value of __sub is zero (the default), in which case this function returns the offset from the beginning of the target sequence to the beginning of the match.

prefix()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
const_reference std::match_results< _Bi_iter, _Alloc >::prefix () const  [inline]
```

Gets a sub_match representing the match prefix.

Precondition

ready() == true

This function gets a reference to a sub_match object representing the part of the target range between the start of the target range and the start of the match.
Referenced by std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const cha

ready()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
bool std::match_results< _Bi_iter, _Alloc >::ready () const  [inline], [noexcept]
```

Indicates if the match_results is ready.

Return values

true	The object has a fully-established result state.
false	The object is not ready.

Referenced by std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const cha
std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const char * >::operator
std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const char * >::prefix(),
and std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const char * >::suffi

size()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
size_type std::match_results< _Bi_iter, _Alloc >::size () const  [inline], [noexcept]
```

Gets the number of matches and submatches.
The number of matches for a given regular expression will be either 0 if there was no match or mark_count() + 1 if a match was successful. Some matches may be empty.

Returns

the number of matches found.

Referenced by std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const cha
and std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const char * >::oper

str()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
string_type std::match_results< _Bi_iter, _Alloc >::str (
              size_type __sub = 0) const  [inline]
```

Gets the match or submatch converted to a string type.

Parameters

__sub	indicates the submatch.

Precondition

ready() == true

This function gets the submatch (or match, if __sub is zero) extracted from the target range and converted to the associated string type.

suffix()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
const_reference std::match_results< _Bi_iter, _Alloc >::suffix () const  [inline]
```

Gets a sub_match representing the match suffix.

Precondition

ready() == true

This function gets a reference to a sub_match object representing the part of the target range between the end of the match and the end of the target range.
Referenced by std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const cha

swap()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
void std::match_results< _Bi_iter, _Alloc >::swap (
            match_results< _Bi_iter, _Alloc > & __that)  [inline], [noexcept]
```

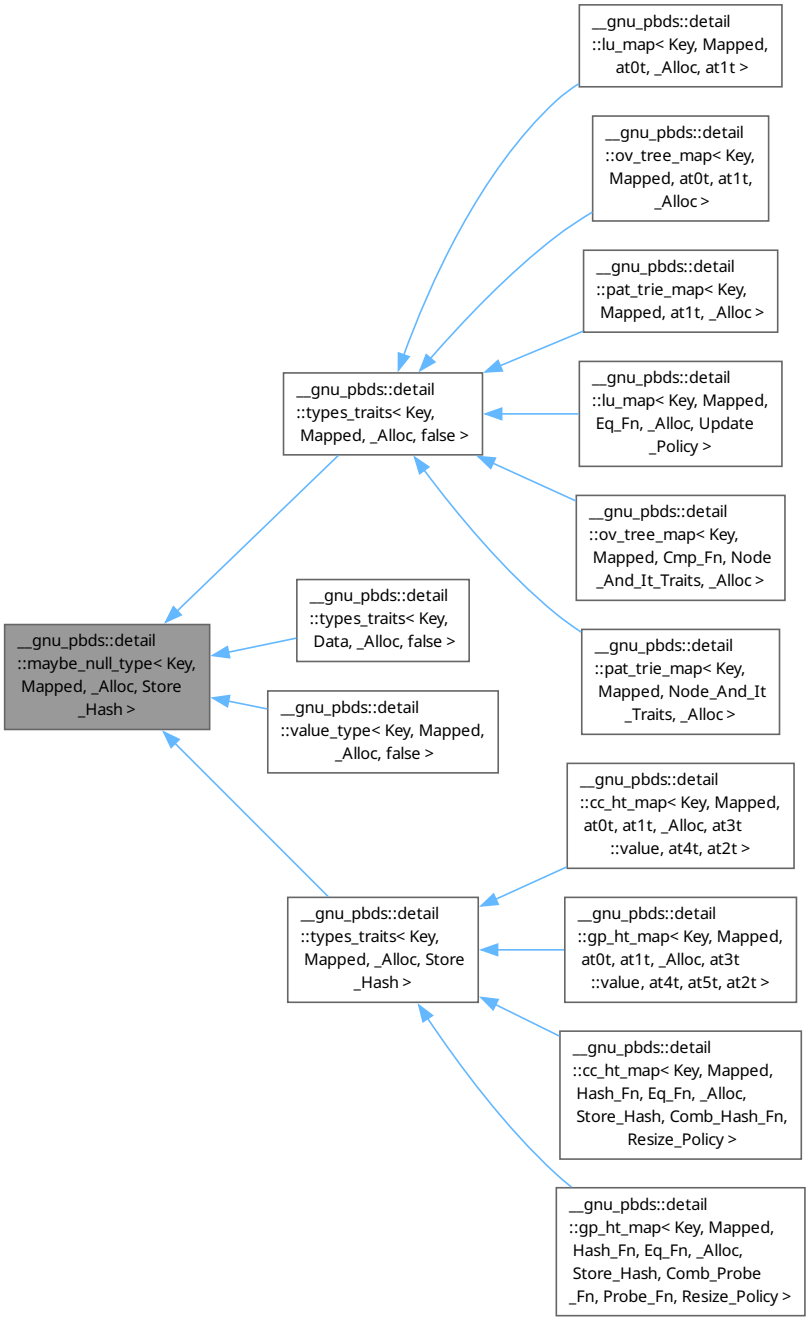
Swaps the contents of two match_results.
Referenced by std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const cha
The documentation for this class was generated from the following files:

- regex.h
- regex.tcc

5.657 __gnu_pbds::detail::maybe_null_type< Key, Mapped, _Alloc, Store_Hash > Struct Template Reference

```
#include <types_traits.hpp>
```

Inheritance diagram for __gnu_pbds::detail::maybe_null_type< Key, Mapped, _Alloc, Store_Hash >:



5.657.1 Detailed Description

template<typename Key, typename Mapped, typename _Alloc, bool Store_Hash>
struct __gnu_pbds::detail::maybe_null_type< Key, Mapped, _Alloc, Store_Hash >

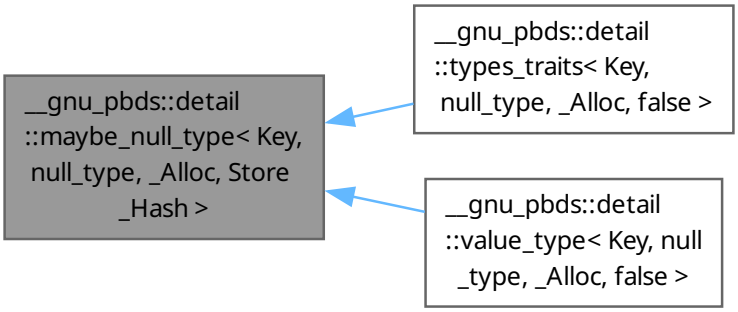
Base class for conditionally defining a static data member.

The documentation for this struct was generated from the following file:

- types_traits.hpp

5.658 __gnu_pbds::detail::maybe_null_type< Key, null_type, _Alloc, Store_Hash > Struct Template Reference

`#include <types_traits.hpp>`
Inheritance diagram for __gnu_pbds::detail::maybe_null_type< Key, null_type, _Alloc, Store_Hash >:



Static Public Attributes

- static null_type **s_null_type**

5.658.1 Detailed Description

template<typename Key, typename _Alloc, bool Store_Hash>
struct __gnu_pbds::detail::maybe_null_type< Key, null_type, _Alloc, Store_Hash >

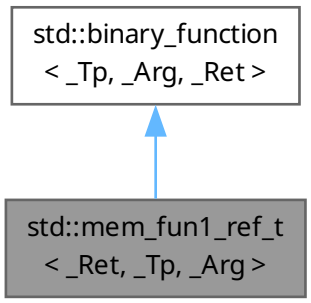
Specialization that defines a static data member of type null_type.
The documentation for this struct was generated from the following file:

- types_traits.hpp

5.659 std::mem_fun1_ref_t< _Ret, _Tp, _Arg > Class Template Reference

`#include <stl_function.h>`

Inheritance diagram for std::mem_fun1_ref_t< _Ret, _Tp, _Arg >:



Public Types

- typedef _Tp first_argument_type
- typedef _Ret result_type
- typedef _Arg second_argument_type

Public Member Functions

- **mem_fun1_ref_t** (_Ret(_Tp::*__pf)(_Arg))
- _Ret **operator()** (_Tp &__r, _Arg __x) const

5.659.1 Detailed Description

template<typename _Ret, typename _Tp, typename _Arg>
class std::mem_fun1_ref_t< _Ret, _Tp, _Arg >

One of the adaptors for member pointers.

5.659.2 Member Typedef Documentation

first_argument_type

```
typedef _Tp std::binary_function< _Tp, _Arg, _Ret >::first_argument_type  [inherited]
```

`first_argument_type` is the type of the first argument

result_type

```
typedef _Ret std::binary_function< _Tp, _Arg, _Ret >::result_type  [inherited]
```

`result_type` is the return type

second_argument_type

```
typedef _Arg std::binary_function< _Tp, _Arg, _Ret >::second_argument_type  [inherited]
```

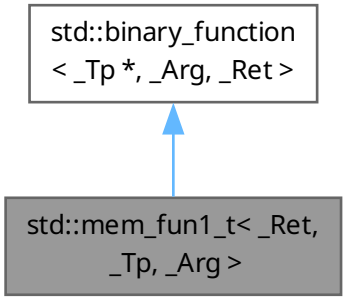
`second_argument_type` is the type of the second argument

The documentation for this class was generated from the following file:

- stl_function.h

5.660 std::mem_fun1_t< _Ret, _Tp, _Arg > Class Template Reference

`#include <stl_function.h>`
Inheritance diagram for std::mem_fun1_t< _Ret, _Tp, _Arg >:



Public Types

- typedef _Tp * first_argument_type
- typedef _Ret result_type
- typedef _Arg second_argument_type

Public Member Functions

- **mem_fun1_t** (_Ret(_Tp::*__pf)(_Arg))
- _Ret **operator()** (_Tp *__p, _Arg __x) const

5.660.1 Detailed Description

template<typename _Ret, typename _Tp, typename _Arg>
class std::mem_fun1_t< _Ret, _Tp, _Arg >

One of the adaptors for member pointers.

5.660.2 Member Typedef Documentation

first_argument_type

`typedef _Tp * std::binary_function< _Tp *, _Arg, _Ret >::first_argument_type [inherited]`
`first_argument_type` is the type of the first argument

result_type

`typedef _Ret std::binary_function< _Tp *, _Arg, _Ret >::result_type [inherited]`
`result_type` is the return type

second_argument_type

```
typedef _Arg std::binary_function< _Tp *, _Arg, _Ret >::second_argument_type  [inherited]
```

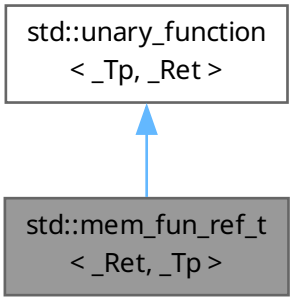
second_argument_type is the type of the second argument
The documentation for this class was generated from the following file:

- stl_function.h

5.661 std::mem_fun_ref_t< _Ret, _Tp > Class Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::mem_fun_ref_t< _Ret, _Tp >:



Public Types

- typedef _Tp argument_type
- typedef _Ret result_type

Public Member Functions

- **mem_fun_ref_t** (_Ret(_Tp::*__pf)())
- _Ret **operator()** (_Tp &__r) const

5.661.1 Detailed Description

template<typename _Ret, typename _Tp>
class std::mem_fun_ref_t< _Ret, _Tp >

One of the adaptors for member pointers.

5.661.2 Member Typedef Documentation

argument_type

```
typedef _Tp std::unary_function< _Tp, _Ret >::argument_type  [inherited]
```

argument_type is the type of the argument

result_type

```
typedef _Ret std::unary_function< _Tp, _Ret >::result_type  [inherited]
```

result_type is the return type

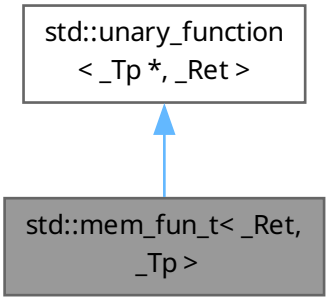
The documentation for this class was generated from the following file:

- stl_function.h

5.662 std::mem_fun_t< _Ret, _Tp > Class Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::mem_fun_t< _Ret, _Tp >:



Public Types

- typedef _Tp * argument_type
- typedef _Ret result_type

Public Member Functions

- **mem_fun_t** (_Ret(_Tp::*__pf)())
- _Ret **operator()** (_Tp *__p) const

5.662.1 Detailed Description

template<typename _Ret, typename _Tp>
class std::mem_fun_t< _Ret, _Tp >

One of the adaptors for member pointers.

5.662.2 Member Typedef Documentation

argument_type

```
typedef _Tp * std::unary_function< _Tp *, _Ret >::argument_type  [inherited]
```

argument_type is the type of the argument

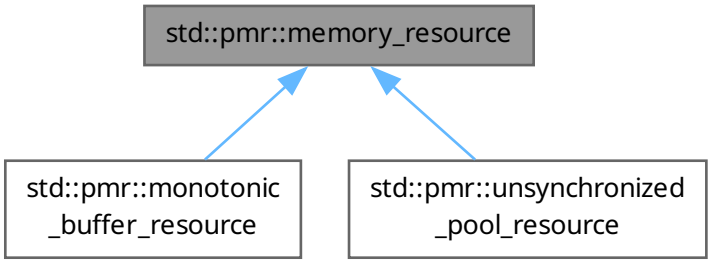
result_type

`typedef _Ret std::unary_function< _Tp *, _Ret >::result_type [inherited]`
`result_type` is the return type
The documentation for this class was generated from the following file:

- stl_function.h

5.663 std::pmr::memory_resource Class Reference

`#include <memory_resource>`
Inheritance diagram for std::pmr::memory_resource:



Public Member Functions

- **memory_resource** (const memory_resource &)=default
- void * **allocate** (size_t __bytes, size_t __alignment=_S_max_align)
- void **deallocate** (void *__p, size_t __bytes, size_t __alignment=_S_max_align)
- bool **is_equal** (const memory_resource &__other) const noexcept
- memory_resource & **operator=** (const memory_resource &)=default

5.663.1 Detailed Description

Class `memory_resource`

Since

C++17

The documentation for this class was generated from the following file:

- memory_resource.h

5.664 std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > Class Template Reference

`#include <random>`

Public Types

- typedef _UIntType result_type

Public Member Functions

- template<typename _Sseq, typename = _If_seed_seq<_Sseq>>
  mersenne_twister_engine (_Sseq &__q)
- **mersenne_twister_engine** (result_type __sd)
- void discard (unsigned long long __z)
- result_type **operator()** ()
- template<typename _Sseq>
  _If_seed_seq< _Sseq > **seed** (_Sseq &__q)
- template<typename _Sseq>
  auto **seed** (_Sseq &__q) -> _If_seed_seq< _Sseq >
- void **seed** (result_type __sd=default_seed)

Static Public Member Functions

- static constexpr result_type max ()
- static constexpr result_type min ()

Static Public Attributes

- static constexpr result_type **default_seed**
- static constexpr result_type **initialization_multiplier**
- static constexpr size_t **mask_bits**
- static constexpr size_t **shift_size**
- static constexpr size_t **state_size**
- static constexpr result_type **tempering_b**
- static constexpr result_type **tempering_c**
- static constexpr result_type **tempering_d**
- static constexpr size_t **tempering_l**
- static constexpr size_t **tempering_s**
- static constexpr size_t **tempering_t**
- static constexpr size_t **tempering_u**
- static constexpr size_t **word_size**
- static constexpr result_type **xor_mask**

Friends

- template<typename _UIntType1, size_t __w1, size_t __n1, size_t __m1, size_t __r1, _UIntType1 __a1, size_t __u1, _UIntType1 __d1, size_t __s1, _UIntType1 __b1, size_t __t1, _UIntType1 __c1, size_t __l1, _UIntType1 __f1, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::mersenne_twister_engine< _UIntType1, __w1, __n1, __m1, __r1, __a1, __u1, __d1, __s1, __b1, __t1, __c1, __l1, __f1 > &__x)
- bool operator== (const mersenne_twister_engine &__lhs, const mersenne_twister_engine &__rhs)
- template<typename _UIntType1, size_t __w1, size_t __n1, size_t __m1, size_t __r1, _UIntType1 __a1, size_t __u1, _UIntType1 __d1, size_t __s1, _UIntType1 __b1, size_t __t1, _UIntType1 __c1, size_t __l1, _UIntType1 __f1, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::mersenne_twister_engine< _UIntType1, __w1, __n1, __m1, __r1, __a1, __u1, __d1, __s1, __b1, __t1, __c1, __l1, __f1 > &__x)

5.664.1 Detailed Description

template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
class std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >

A generalized feedback shift register discrete random number generator.
This algorithm avoids multiplication and division and is designed to be friendly to a pipelined architecture. If the parameters are chosen correctly, this generator will produce numbers with a very long period and fairly good apparent entropy, although still not cryptographically strong.
The best way to use this generator is with the predefined mt19937 class.
This algorithm was originally invented by Makoto Matsumoto and Takuji Nishimura.

Template Parameters

_w	Word size, the number of bits in each element of the state vector.
_n	The degree of recursion.
_m	The period parameter.
_r	The separation point bit index.
_a	The last row of the twist matrix.
_u	The first right-shift tempering matrix parameter.
_d	The first right-shift tempering matrix mask.
_s	The first left-shift tempering matrix parameter.
_b	The first left-shift tempering matrix mask.
_t	The second left-shift tempering matrix parameter.
_c	The second left-shift tempering matrix mask.
_l	The second right-shift tempering matrix parameter.
_f	Initialization multiplier.

Since

C++11

5.664.2 Member Typedef Documentation

result_type

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
```

typedef _UIntType std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >::result_type
The type of the generated random value.

5.664.3 Constructor & Destructor Documentation

mersenne_twister_engine()

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
template<typename _Sseq, typename = _If_seed_seq<_Sseq>>
std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >::mersenne_twister_engine (
            _Sseq & __q)  [inline], [explicit]
```

Constructs a mersenne_twister_engine random number generator engine seeded from the seed sequence __q.

Parameters

_q	the seed sequence.

5.664.4 Member Function Documentation

discard()

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
void std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >::discard (
            unsigned long long __z)
```

Discard a sequence of random numbers.

max()

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
static constexpr result_type std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >::max ()  [inline], [static], [constexpr]
```

Gets the largest possible value in the output range.

min()

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
static constexpr result_type std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >::min ()  [inline], [static], [constexpr]
```

Gets the smallest possible value in the output range.

5.664.5 Friends And Related Symbol Documentation

operator<<

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size↩
_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType
__f>
template<typename _UIntType1, size_t __w1, size_t __n1, size_t __m1, size_t __r1, _UIntType1 ↩
__a1, size_t __u1, _UIntType1 __d1, size_t __s1, _UIntType1 __b1, size_t __t1, _UIntType1 __c1,
size_t __l1, _UIntType1 __f1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::mersenne_twister_engine< _UIntType1, __w1, __n1, __m1, __r1, __a1, __u1,
__d1, __s1, __b1, __t1, __c1, __l1, __f1 > & __x)  [friend]
```

Inserts the current state of a % mersenne_twister_engine random number generator engine ___x into the output stream ___os.

Parameters

__os	An output stream.
__x	A % mersenne_twister_engine random number generator engine.

Returns

The output stream with the state of ___x inserted or in an error state.

operator==

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size↩
_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType
__f>
bool operator== (
            const mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, _↩
_b, __t, __c, __l, __f > & __lhs,
            const mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, _↩
_b, __t, __c, __l, __f > & __rhs)  [friend]
```

Compares two % mersenne_twister_engine random number generator objects of the same type for equality.

Parameters

__lhs	A % mersenne_twister_engine random number generator object.
__rhs	Another % mersenne_twister_engine random number generator object.

Returns

true if the infinite sequences of generated values would be equal, false otherwise.

operator>>

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size↩
_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType
__f>
```

```
template<typename _UIntType1, size_t __w1, size_t __n1, size_t __m1, size_t __r1, _UIntType1 ↩
__a1, size_t __u1, _UIntType1 __d1, size_t __s1, _UIntType1 __b1, size_t __t1, _UIntType1 __c1,
size_t __l1, _UIntType1 __f1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            std::mersenne_twister_engine< _UIntType1, __w1, __n1, __m1, __r1, __a1, __u1, __d1,
__s1, __b1, __t1, __c1, __l1, __f1 > & __x)  [friend]
```

Extracts the current state of a % mersenne_twister_engine random number generator engine __x from the input stream __is.

Parameters

_↩ _is	An input stream.
_↩ _x	A % mersenne_twister_engine random number generator engine.

Returns

The input stream with the state of __x extracted or in an error state.

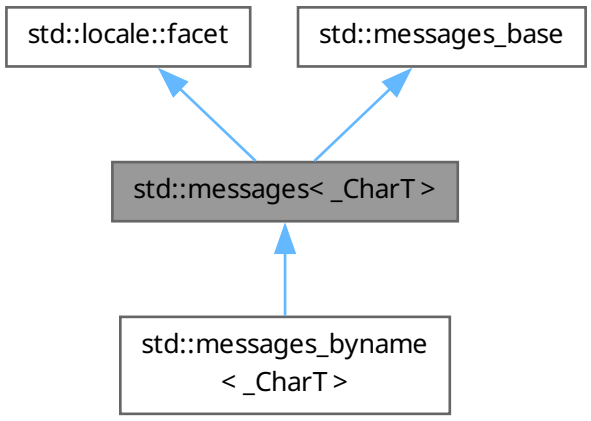
The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

5.665 std::messages< _CharT > Class Template Reference

`#include <locale_facets_nonio.h>`

Inheritance diagram for std::messages< _CharT >:



Public Types

- typedef int **catalog**

- typedef _CharT char_type
- typedef basic_string< _CharT > string_type

Public Member Functions

- messages (__c_locale __cloc, const char *__s, size_t __refs=0)
- messages (size_t __refs=0)
- void **close** (catalog __c) const
- string_type **get** (catalog __c, int __set, int __msgid, const string_type &__s) const
- catalog **open** (const basic_string< char > &, const locale &, const char *) const
- catalog **open** (const basic_string< char > &__s, const locale &__loc) const

Static Public Attributes

- static locale::id id

Protected Member Functions

- virtual ~messages ()
- string_type **_M_convert_from_char** (char *) const
- char * **_M_convert_to_char** (const string_type &__msg) const
- virtual void **do_close** (catalog) const
- void **do_close** (catalog) const
- void **do_close** (catalog) const
- string do_get (catalog, int, int, const string &) const
- virtual string_type **do_get** (catalog, int, int, const string_type &__dfault) const
- wstring **do_get** (catalog, int, int, const wstring &) const
- virtual catalog **do_open** (const basic_string< char > &, const locale &) const
- messages< char >::catalog **do_open** (const basic_string< char > &, const locale &) const
- messages< wchar_t >::catalog **do_open** (const basic_string< char > &, const locale &) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale **_M_c_locale_messages**
- const char * **_M_name_messages**

5.665.1 Detailed Description

template<typename _CharT>
class std::messages< _CharT >

Primary class template messages.
This facet encapsulates the code to retrieve messages from message catalogs. The only thing defined by the standard for this facet is the interface. All underlying functionality is implementation-defined.

This library currently implements 3 versions of the message facet. The first version (gnu) is a wrapper around gettext, provided by libintl. The second version (ieee) is a wrapper around catgets. The final version (default) does no actual translation. These implementations are only provided for char and wchar_t instantiations.
The messages template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the messages facet.

5.665.2 Member Typedef Documentation

char_type

```
template<typename _CharT>
typedef _CharT std::messages< _CharT >::char_type
```

Public typedefs.

string_type

```
template<typename _CharT>
typedef basic_string<_CharT> std::messages< _CharT >::string_type
```

Public typedefs.

5.665.3 Constructor & Destructor Documentation

messages() [1/2]

```
template<typename _CharT>
std::messages< _CharT >::messages (
            size_t __refs = 0)  [explicit]
```

Constructor performs initialization.
This is the constructor provided by the standard.

Parameters

__refs	Passed to the base facet class.

References std::locale::facet::facet().

messages() [2/2]

```
template<typename _CharT>
std::messages< _CharT >::messages (
            __c_locale __cloc,
            const char * __s,
            size_t __refs = 0)  [explicit]
```

Internal constructor. Not for general use.
This is a constructor for use by the library itself to set up new locales.

Parameters

__cloc	The C locale.
__s	The name of a locale.
__refs	Refcount to pass to the base class.

References std::locale::facet::facet().

~messages()

```
template<typename _CharT>
std::messages< _CharT >::~messages ()  [protected], [virtual]
```

Destructor.

5.665.4 Member Function Documentation

do_get()

```
string std::messages< char >::do_get (
            catalog ,
            int ,
            int ,
            const string & ) const  [protected]
```

Specializations for required instantiations.

5.665.5 Member Data Documentation

id

```
template<typename _CharT>
locale::id std::messages< _CharT >::id  [static]
```

Numpunct facet id.

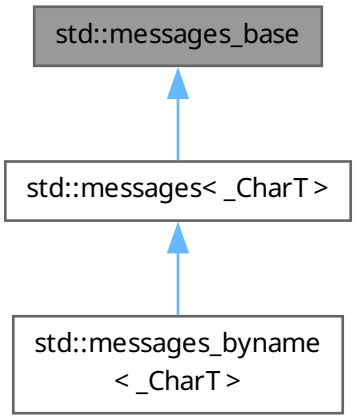
The documentation for this class was generated from the following files:

- locale_facets_nonio.h
- messages_members.h

5.666 std::messages_base Struct Reference

```
#include <locale_facets_nonio.h>
```

Inheritance diagram for std::messages_base:



Public Types

- typedef int **catalog**

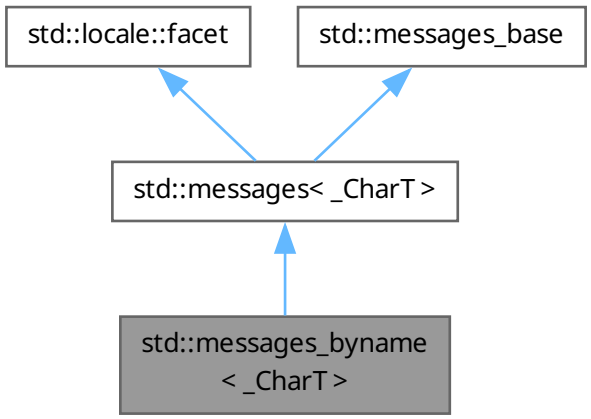
5.666.1 Detailed Description

Messages facet base class providing catalog typedef.
The documentation for this struct was generated from the following file:

- locale_facets_nonio.h

5.667 std::messages_byname< _CharT > Class Template Reference

`#include <locale_facets_nonio.h>`
Inheritance diagram for std::messages_byname< _CharT >:



Public Types

- typedef int **catalog**
- typedef _CharT **char_type**
- typedef basic_string< _CharT > **string_type**

Public Member Functions

- **messages_byname** (const char *__s, size_t __refs=0)
- **messages_byname** (const string &__s, size_t __refs=0)
- void **close** (catalog __c) const
- string_type **get** (catalog __c, int __set, int __msgid, const string_type &__s) const
- catalog **open** (const basic_string< char > &, const locale &, const char *) const
- catalog **open** (const basic_string< char > &__s, const locale &__loc) const

Static Public Attributes

- static locale::id id

Protected Member Functions

- string_type **_M_convert_from_char** (char ∗) const
- char ∗ **_M_convert_to_char** (const string_type &__msg) const
- virtual void **do_close** (catalog) const
- void **do_close** (catalog) const
- void **do_close** (catalog) const
- string do_get (catalog, int, int, const string &) const
- virtual string_type **do_get** (catalog, int, int, const string_type &__dfault) const
- wstring **do_get** (catalog, int, int, const wstring &) const
- virtual catalog **do_open** (const basic_string< char > &, const locale &) const
- messages< char >::catalog **do_open** (const basic_string< char > &, const locale &) const
- messages< wchar_t >::catalog **do_open** (const basic_string< char > &, const locale &) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char ∗__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char ∗ **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char ∗__s)

Protected Attributes

- __c_locale **_M_c_locale_messages**
- const char ∗ **_M_name_messages**

5.667.1 Detailed Description

template<typename _CharT>
class std::messages_byname< _CharT >

class messages_byname [22.2.7.2].

5.667.2 Member Function Documentation

do_get()

```
string std::messages< char >::do_get (
            catalog ,
            int ,
            int ,
            const string & ) const  [protected], [inherited]
```

Specializations for required instantiations.

5.667.3 Member Data Documentation

id

```
template<typename _CharT>
locale::id std::messages< _CharT >::id  [static], [inherited]
```

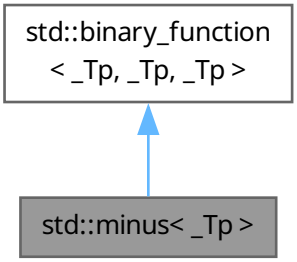
Numpunct facet id.

The documentation for this class was generated from the following files:

- locale_facets_nonio.h
- messages_members.h

5.668 std::minus< _Tp > Struct Template Reference

`#include <stl_function.h>`
Inheritance diagram for std::minus< _Tp >:



Public Types

- typedef _Tp first_argument_type
- typedef _Tp result_type
- typedef _Tp second_argument_type

Public Member Functions

- constexpr _Tp **operator()** (const _Tp &__x, const _Tp &__y) const

5.668.1 Detailed Description

template<typename _Tp>
struct std::minus< _Tp >

One of the math functors.

5.668.2 Member Typedef Documentation

first_argument_type

`typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::first_argument_type [inherited]`
`first_argument_type` is the type of the first argument

result_type

`typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::result_type [inherited]`
`result_type` is the return type

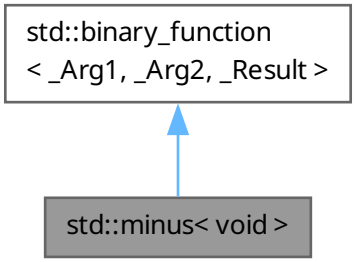
second_argument_type

`typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::second_argument_type [inherited]`
`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

5.669 std::minus< void > Struct Reference

`#include <stl_function.h>`
Inheritance diagram for std::minus< void >:



Public Types

- typedef void first_argument_type
- typedef __is_transparent **is_transparent**
- typedef void result_type
- typedef void second_argument_type

Public Member Functions

- template<typename _Tp, typename _Up>
  constexpr auto **operator()** (_Tp &&__t, _Up &&__u) const noexcept(noexcept(std::forward< _Tp >(__t) - std::forward< _Up >(__u))) -> decltype(std::forward< _Tp >(__t) - std::forward< _Up >(__u))
- constexpr void **operator()** (const void &__x, const void &__y) const

5.669.1 Detailed Description

One of the math functors.

5.669.2 Member Typedef Documentation

first_argument_type

`typedef void std::binary_function< void, void, void >::first_argument_type`
`first_argument_type` is the type of the first argument

result_type

`typedef void std::binary_function< void, void, void >::result_type`
`result_type` is the return type

second_argument_type

`typedef void std::binary_function< void, void, void >::second_argument_type`
`second_argument_type` is the type of the second argument

The documentation for this struct was generated from the following file:

- stl_function.h

5.670 __gnu_pbds::detail::mod_based_range_hashing< Size_Type > Class Template Reference

`#include <mod_based_range_hashing.hpp>`

Protected Types

- typedef Size_Type **size_type**

Protected Member Functions

- void **notify_resized** (size_type s)
- size_type **range_hash** (size_type s) const
- void **swap** (mod_based_range_hashing &other)

5.670.1 Detailed Description

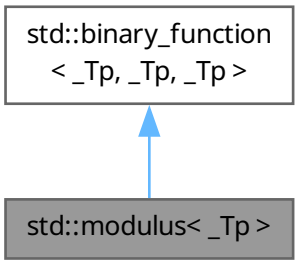
template<typename Size_Type>
class __gnu_pbds::detail::mod_based_range_hashing< Size_Type >

Mod based range hashing.
The documentation for this class was generated from the following file:

- mod_based_range_hashing.hpp

5.671 std::modulus< _Tp > Struct Template Reference

`#include <stl_function.h>`
Inheritance diagram for std::modulus< _Tp >:



Public Types

- typedef _Tp first_argument_type
- typedef _Tp result_type
- typedef _Tp second_argument_type

Public Member Functions

- constexpr _Tp **operator()** (const _Tp &__x, const _Tp &__y) const

5.671.1 Detailed Description

template<**typename _Tp**>
struct std::modulus< **_Tp** >

One of the math functors.

5.671.2 Member Typedef Documentation

first_argument_type

`typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::first_argument_type [inherited]`
`first_argument_type` is the type of the first argument

result_type

`typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::result_type [inherited]`
`result_type` is the return type

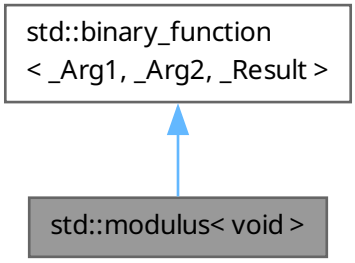
second_argument_type

`typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::second_argument_type [inherited]`
`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

5.672 std::modulus< void > Struct Reference

`#include <stl_function.h>`
Inheritance diagram for std::modulus< void >:



Public Types

- typedef void first_argument_type
- typedef __is_transparent **is_transparent**

- typedef void result_type
- typedef void second_argument_type

Public Member Functions

- template<typename _Tp, typename _Up>
  constexpr auto **operator()** (_Tp &&__t, _Up &&__u) const noexcept(noexcept(std::forward< _Tp >(__t) % std::forward< _Up >(__u))) -> decltype(std::forward< _Tp >(__t) % std::forward< _Up >(__u))
- constexpr void **operator()** (const void &__x, const void &__y) const

5.672.1 Detailed Description

One of the math functors.

5.672.2 Member Typedef Documentation

first_argument_type

`typedef void std::binary_function< void, void, void >::first_argument_type`
`first_argument_type` is the type of the first argument

result_type

`typedef void std::binary_function< void, void, void >::result_type`
`result_type` is the return type

second_argument_type

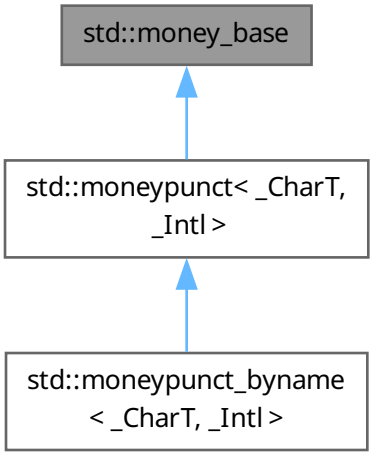
`typedef void std::binary_function< void, void, void >::second_argument_type`
`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

5.673 std::money_base Class Reference

`#include <locale_facets_nonio.h>`

Inheritance diagram for std::money_base:



Public Types

- enum { **_S_minus** , **_S_zero** , **_S_end** }
- enum **part** {
  none , **space** , **symbol** , **sign** ,
  value }

Static Public Member Functions

- static pattern **_S_construct_pattern** (char __precedes, char __space, char __posn) throw ()

Static Public Attributes

- static const char * **_S_atoms**
- static const pattern **_S_default_pattern**

5.673.1 Detailed Description

Money format ordering data.
This class contains an ordered array of 4 fields to represent the pattern for formatting a money amount. Each field may contain one entry from the part enum. symbol, sign, and value must be present and the remaining field must contain either none or space.

See also

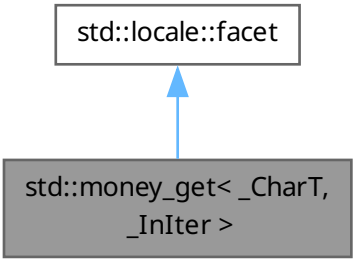
moneypunct::pos_format() and moneypunct::neg_format() for details of how these fields are interpreted.

The documentation for this class was generated from the following file:

- locale_facets_nonio.h

5.674 std::money_get< _CharT, _InIter > Class Template Reference

`#include <locale_facets_nonio.h>`
Inheritance diagram for std::money_get< _CharT, _InIter >:



Public Types

- typedef _CharT char_type
- typedef _InIter iter_type
- typedef basic_string< _CharT > string_type

Public Member Functions

- money_get (size_t __refs=0)
- template<bool _Intl>
  _GLIBCXX_BEGIN_NAMESPACE_LDBL_OR_CXX11 _InIter **_M_extract** (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, string &__units) const
- iter_type get (iter_type __s, iter_type __end, bool __intl, ios_base &__io, ios_base::iostate &__err, long double &__units) const
- iter_type get (iter_type __s, iter_type __end, bool __intl, ios_base &__io, ios_base::iostate &__err, string_type &__digits) const

Static Public Attributes

- static locale::id id

Protected Member Functions

- virtual ~money_get ()
- template<bool _Intl>
  iter_type **_M_extract** (iter_type __s, iter_type __end, ios_base &__io, ios_base::iostate &__err, string &__digits) const
- virtual iter_type do_get (iter_type __s, iter_type __end, bool __intl, ios_base &__io, ios_base::iostate &__err, long double &__units) const
- virtual iter_type do_get (iter_type __s, iter_type __end, bool __intl, ios_base &__io, ios_base::iostate &__err, string_type &__digits) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

5.674.1 Detailed Description

template<typename _CharT, typename _InIter>
class std::money_get< _CharT, _InIter >

Primary class template money_get.
This facet encapsulates the code to parse and return a monetary amount from a string.
The money_get template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the money_get facet.

5.674.2 Member Typedef Documentation

char_type

```
template<typename _CharT, typename _InIter>
typedef _CharT std::money_get< _CharT, _InIter >::char_type
```

Public typedefs.

iter_type

```
template<typename _CharT, typename _InIter>
typedef _InIter std::money_get< _CharT, _InIter >::iter_type
```

Public typedefs.

string_type

```
template<typename _CharT, typename _InIter>
typedef basic_string<_CharT> std::money_get< _CharT, _InIter >::string_type
```

Public typedefs.

5.674.3 Constructor & Destructor Documentation

money_get()

```
template<typename _CharT, typename _InIter>
std::money_get< _CharT, _InIter >::money_get (
            size_t __refs = 0)  [inline], [explicit]
```

Constructor performs initialization.
This is the constructor provided by the standard.

Parameters

__refs	Passed to the base facet class.

References std::locale::facet::facet().

~money_get()

```
template<typename _CharT, typename _InIter>
virtual std::money_get< _CharT, _InIter >::~money_get ()  [inline], [protected], [virtual]
```

Destructor.

5.674.4 Member Function Documentation

do_get() [1/2]

```
template<typename _CharT, typename _InIter>
_InIter std::money_get< _CharT, _InIter >::do_get (
            iter_type __s,
            iter_type __end,
            bool __intl,
            ios_base & __io,
            ios_base::iostate & __err,
            long double & __units) const  [protected], [virtual]
```

Read and parse a monetary value.

This function reads and parses characters representing a monetary value. This function is a hook for derived classes to change the value returned.

See also

get() for details.

References std::basic_string< _CharT, _Traits, _Alloc >::c_str().

Referenced by get(), and get().

do_get() [2/2]

```
template<typename _CharT, typename _InIter>
_InIter std::money_get< _CharT, _InIter >::do_get (
            iter_type __s,
            iter_type __end,
            bool __intl,
            ios_base & __io,
            ios_base::iostate & __err,
            string_type & __digits) const  [protected], [virtual]
```

Read and parse a monetary value.

This function reads and parses characters representing a monetary value. This function is a hook for derived classes to change the value returned.

See also

get() for details.

References std::ios_base::_M_getloc(), std::use_facet(), and std::__ctype_abstract_base< _CharT >::widen().

get() [1/2]

```
template<typename _CharT, typename _InIter>
iter_type std::money_get< _CharT, _InIter >::get (
            iter_type __s,
            iter_type __end,
            bool __intl,
            ios_base & __io,
```

```
            ios_base::iostate & __err,
            long double & __units) const  [inline]
```

Read and parse a monetary value.

This function reads characters from *__s*, interprets them as a monetary value according to moneypunct and ctype facets retrieved from io.getloc(), and returns the result in *units* as an integral value moneypunct::frac_digits() * the actual amount. For example, the string $10.01 in a US locale would store 1001 in *units*.

Any characters not part of a valid money amount are not consumed.

If a money value cannot be parsed from the input stream, sets err=(err|io.failbit). If the stream is consumed before finishing parsing, sets err=(err|io.failbit|io.eofbit). *units* is unchanged if parsing fails.

This function works by returning the result of do_get().

Parameters

__s	Start of characters to parse.
__end	End of characters to parse.
__intl	Parameter to use_facet<moneypunct<CharT,intl> >.
__io	Source of facets and io state.
__err	Error field to set if parsing fails.
__units	Place to store result of parsing.

Returns

Iterator referencing first character beyond valid money amount.

References do_get().

get() [2/2]

```
template<typename _CharT, typename _InIter>
iter_type std::money_get< _CharT, _InIter >::get (
            iter_type __s,
            iter_type __end,
            bool __intl,
            ios_base & __io,
            ios_base::iostate & __err,
            string_type & __digits) const  [inline]
```

Read and parse a monetary value.

This function reads characters from *__s*, interprets them as a monetary value according to moneypunct and ctype facets retrieved from io.getloc(), and returns the result in *digits*. For example, the string $10.01 in a US locale would store `1001` in *digits*.

Any characters not part of a valid money amount are not consumed.

If a money value cannot be parsed from the input stream, sets err=(err|io.failbit). If the stream is consumed before finishing parsing, sets err=(err|io.failbit|io.eofbit).

This function works by returning the result of do_get().

Parameters

__s	Start of characters to parse.
__end	End of characters to parse.
__intl	Parameter to use_facet<moneypunct<CharT,intl> >.
__io	Source of facets and io state.
__err	Error field to set if parsing fails.
__digits	Place to store result of parsing.

Returns

Iterator referencing first character beyond valid money amount.

References do_get().

5.674.5 Member Data Documentation

id

```
template<typename _CharT, typename _InIter>
locale::id std::money_get< _CharT, _InIter >::id  [static]
```

Numpunct facet id.

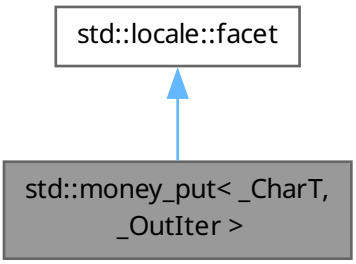
The documentation for this class was generated from the following files:

- locale_facets_nonio.h
- locale_facets_nonio.tcc

5.675 std::money_put< _CharT, _OutIter > Class Template Reference

`#include <locale_facets_nonio.h>`

Inheritance diagram for std::money_put< _CharT, _OutIter >:



Public Types

- typedef _CharT char_type
- typedef _OutIter iter_type
- typedef basic_string< _CharT > string_type

Public Member Functions

- money_put (size_t __refs=0)
- template<bool _Intl>
  _OutIter **_M_insert** (iter_type __s, ios_base &__io, char_type __fill, const string_type &__digits) const
- iter_type put (iter_type __s, bool __intl, ios_base &__io, char_type __fill, const string_type &__digits) const
- iter_type put (iter_type __s, bool __intl, ios_base &__io, char_type __fill, long double __units) const

Static Public Attributes

- static locale::id id

Protected Member Functions

- virtual ~money_put ()
- template<bool _Intl>
  iter_type **_M_insert** (iter_type __s, ios_base &__io, char_type __fill, const string_type &__digits) const
- virtual iter_type do_put (iter_type __s, bool __intl, ios_base &__io, char_type __fill, const string_type &__digits) const
- virtual iter_type do_put (iter_type __s, bool __intl, ios_base &__io, char_type __fill, long double __units) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

5.675.1 Detailed Description

template<typename _CharT, typename _OutIter>
class std::money_put< _CharT, _OutIter >

Primary class template money_put.
This facet encapsulates the code to format and output a monetary amount.
The money_put template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the money_put facet.

5.675.2 Member Typedef Documentation

char_type

```
template<typename _CharT, typename _OutIter>
typedef _CharT std::money_put< _CharT, _OutIter >::char_type
```

Public typedefs.

iter_type

```
template<typename _CharT, typename _OutIter>
typedef _OutIter std::money_put< _CharT, _OutIter >::iter_type
```

Public typedefs.

string_type

```
template<typename _CharT, typename _OutIter>
typedef basic_string<_CharT> std::money_put< _CharT, _OutIter >::string_type
```

Public typedefs.

5.675.3 Constructor & Destructor Documentation

money_put()

```
template<typename _CharT, typename _OutIter>
std::money_put< _CharT, _OutIter >::money_put (
            size_t __refs = 0)  [inline], [explicit]
```

Constructor performs initialization.
This is the constructor provided by the standard.

Parameters

__refs	Passed to the base facet class.

References std::locale::facet::facet().

~money_put()

```
template<typename _CharT, typename _OutIter>
virtual std::money_put< _CharT, _OutIter >::~money_put ()  [inline], [protected], [virtual]
```

Destructor.

5.675.4 Member Function Documentation

do_put() [1/2]

```
template<typename _CharT, typename _OutIter>
_OutIter std::money_put< _CharT, _OutIter >::do_put (
            iter_type __s,
            bool __intl,
            ios_base & __io,
            char_type __fill,
            const string_type & __digits) const  [protected], [virtual]
```

Format and output a monetary value.
This function formats *digits* as a monetary value according to moneypunct and ctype facets retrieved from io.getloc(), and writes the resulting characters to *__s*. For example, the string `1001` in a US locale would write `$10.01` to *__s*.
This function is a hook for derived classes to change the value returned.

See also

put().

Parameters

__s	The stream to write to.
__intl	Parameter to use_facet<moneypunct<CharT,intl> >.
__io	Source of facets and io state.
__fill	char_type to use for padding.
__digits	Place to store result of parsing.

Returns

Iterator after writing.

do_put() [2/2]

```
template<typename _CharT, typename _OutIter>
_OutIter std::money_put< _CharT, _OutIter >::do_put (
            iter_type __s,
            bool __intl,
            ios_base & __io,
            char_type __fill,
            long double __units) const  [protected], [virtual]
```

Format and output a monetary value.
This function formats *units* as a monetary value according to moneypunct and ctype facets retrieved from io.getloc(), and writes the resulting characters to *__s*. For example, the value 1001 in a US locale would write `$10.01` to *__s*.
This function is a hook for derived classes to change the value returned.

See also

put().

Parameters

__s	The stream to write to.
__intl	Parameter to use_facet<moneypunct<CharT,intl> >.
__io	Source of facets and io state.
__fill	char_type to use for padding.
__units	Place to store result of parsing.

Returns

Iterator after writing.

References std::ios_base::getloc(), std::use_facet(), and std::__ctype_abstract_base< _CharT >::widen().
Referenced by put(), and put().

put() [1/2]

```
template<typename _CharT, typename _OutIter>
iter_type std::money_put< _CharT, _OutIter >::put (
            iter_type __s,
            bool __intl,
            ios_base & __io,
            char_type __fill,
            const string_type & __digits) const  [inline]
```

Format and output a monetary value.
This function formats *digits* as a monetary value according to moneypunct and ctype facets retrieved from io.getloc(), and writes the resulting characters to *__s*. For example, the string `1001` in a US locale would write `$10.01` to *__s*.
This function works by returning the result of do_put().

Parameters

__s	The stream to write to.
__intl	Parameter to use_facet<moneypunct<CharT,intl> >.
__io	Source of facets and io state.
__fill	char_type to use for padding.
__digits	Place to store result of parsing.

Returns

Iterator after writing.

References do_put().

put() [2/2]

```
template<typename _CharT, typename _OutIter>
iter_type std::money_put< _CharT, _OutIter >::put (
            iter_type __s,
            bool __intl,
            ios_base & __io,
            char_type __fill,
            long double __units) const  [inline]
```

Format and output a monetary value.
This function formats *units* as a monetary value according to moneypunct and ctype facets retrieved from io.getloc(), and writes the resulting characters to *__s*. For example, the value 1001 in a US locale would write `$10.01` to *__s*.
This function works by returning the result of do_put().

Parameters

__s	The stream to write to.
__intl	Parameter to use_facet<moneypunct<CharT,intl> >.
__io	Source of facets and io state.
__fill	char_type to use for padding.
__units	Place to store result of parsing.

Returns

Iterator after writing.

References do_put().

5.675.5 Member Data Documentation

id

```
template<typename _CharT, typename _OutIter>
locale::id std::money_put< _CharT, _OutIter >::id  [static]
```

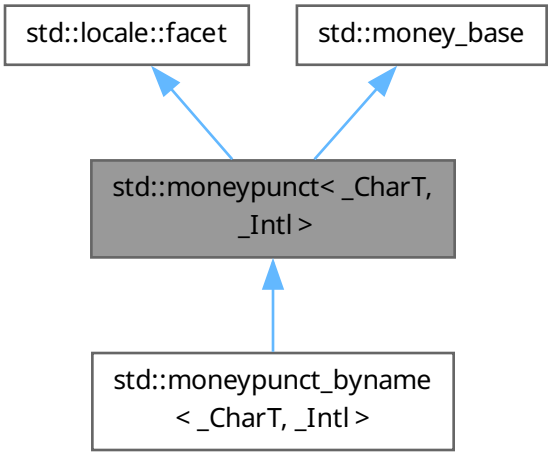
Numpunct facet id.
The documentation for this class was generated from the following files:

- locale_facets_nonio.h
- locale_facets_nonio.tcc

5.676 std::moneypunct< _CharT, _Intl > Class Template Reference

```
#include <locale_facets_nonio.h>
```

Inheritance diagram for std::moneypunct< _CharT, _Intl >:



Public Types

- enum { **_S_minus** , **_S_zero** , **_S_end** }
- typedef __moneypunct_cache< _CharT, _Intl > **__cache_type**
- enum **part** {
  none , **space** , **symbol** , **sign** ,
  value }

- typedef _CharT char_type
- typedef basic_string< _CharT > string_type

Public Member Functions

- moneypunct (__c_locale __cloc, const char *__s, size_t __refs=0)
- moneypunct (__cache_type *__cache, size_t __refs=0)
- moneypunct (size_t __refs=0)
- string_type curr_symbol () const
- char_type decimal_point () const
- int frac_digits () const
- string grouping () const
- string_type negative_sign () const
- string_type positive_sign () const
- char_type thousands_sep () const

- pattern pos_format () const
- pattern neg_format () const

Static Public Member Functions

- static pattern **_S_construct_pattern** (char __precedes, char __space, char __posn) throw ()

Static Public Attributes

- static const char * **_S_atoms**
- static const pattern **_S_default_pattern**
- static locale::id id
- static const bool intl

Protected Member Functions

- virtual ~moneypunct ()
- void **_M_initialize_moneypunct** (__c_locale __cloc=0, const char *__name=0)
- void **_M_initialize_moneypunct** (__c_locale, const char *)
- void **_M_initialize_moneypunct** (__c_locale, const char *)
- void **_M_initialize_moneypunct** (__c_locale, const char *)
- void **_M_initialize_moneypunct** (__c_locale, const char *)
- virtual string_type do_curr_symbol () const
- virtual char_type do_decimal_point () const
- virtual int do_frac_digits () const
- virtual string do_grouping () const
- virtual pattern do_neg_format () const
- virtual string_type do_negative_sign () const
- virtual pattern do_pos_format () const
- virtual string_type do_positive_sign () const
- virtual char_type do_thousands_sep () const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

5.676.1 Detailed Description

template<typename _CharT, bool _Intl>
class std::moneypunct< _CharT, _Intl >

Primary class template moneypunct.
This facet encapsulates the punctuation, grouping and other formatting features of money amount string representations.

5.676.2 Member Typedef Documentation

char_type

```
template<typename _CharT, bool _Intl>
typedef _CharT std::moneypunct< _CharT, _Intl >::char_type
```

Public typedefs.

string_type

```
template<typename _CharT, bool _Intl>
typedef basic_string<_CharT> std::moneypunct< _CharT, _Intl >::string_type
```

Public typedefs.

5.676.3 Constructor & Destructor Documentation

moneypunct() [1/3]

```
template<typename _CharT, bool _Intl>
std::moneypunct< _CharT, _Intl >::moneypunct (
            size_t __refs = 0)  [inline], [explicit]
```

Constructor performs initialization.
This is the constructor provided by the standard.

Parameters

__refs	Passed to the base facet class.

References std::locale::facet::facet().

moneypunct() [2/3]

```
template<typename _CharT, bool _Intl>
std::moneypunct< _CharT, _Intl >::moneypunct (
            __cache_type * __cache,
            size_t __refs = 0)  [inline], [explicit]
```

Constructor performs initialization.
This is an internal constructor.

Parameters

__cache	Cache for optimization.
__refs	Passed to the base facet class.

References std::locale::facet::facet().

moneypunct() [3/3]

```
template<typename _CharT, bool _Intl>
std::moneypunct< _CharT, _Intl >::moneypunct (
            __c_locale __cloc,
            const char * __s,
            size_t __refs = 0)  [inline], [explicit]
```

Internal constructor. Not for general use.
This is a constructor for use by the library itself to set up new locales.

Parameters

__cloc	The C locale.
__s	The name of a locale.
__refs	Passed to the base facet class.

References std::locale::facet::facet().

~moneypunct()

```
template<typename _CharT, bool _Intl>
virtual std::moneypunct< _CharT, _Intl >::~moneypunct ()  [protected], [virtual]
```

Destructor.

5.676.4 Member Function Documentation

curr_symbol()

```
template<typename _CharT, bool _Intl>
string_type std::moneypunct< _CharT, _Intl >::curr_symbol () const  [inline]
```

Return currency symbol string.

This function returns a string_type to use as a currency symbol. It does so by returning returning moneypunct<char_type>::do_curr_symbol().

Returns

string_type representing a currency symbol.

References do_curr_symbol().

decimal_point()

```
template<typename _CharT, bool _Intl>
char_type std::moneypunct< _CharT, _Intl >::decimal_point () const  [inline]
```

Return decimal point character.

This function returns a char_type to use as a decimal point. It does so by returning returning moneypunct<char_type>::do_decimal_point().

Returns

char_type representing a decimal point.

References do_decimal_point().

do_curr_symbol()

```
template<typename _CharT, bool _Intl>
virtual string_type std::moneypunct< _CharT, _Intl >::do_curr_symbol () const  [inline], [protected],
[virtual]
```

Return currency symbol string.

This function returns a string_type to use as a currency symbol. This function is a hook for derived classes to change the value returned.

See also

curr_symbol() for details.

Returns

string_type representing a currency symbol.

Referenced by curr_symbol().

do_decimal_point()

```
template<typename _CharT, bool _Intl>
virtual char_type std::moneypunct< _CharT, _Intl >::do_decimal_point () const  [inline], [protected],
[virtual]
```

Return decimal point character.

Returns a char_type to use as a decimal point. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a decimal point.

Referenced by decimal_point().

do_frac_digits()

```
template<typename _CharT, bool _Intl>
virtual int std::moneypunct< _CharT, _Intl >::do_frac_digits () const  [inline], [protected],
[virtual]
```

Return number of digits in fraction.
This function returns the exact number of digits that make up the fractional part of a money amount. This function is a hook for derived classes to change the value returned.

See also

frac_digits() for details.

Returns

Number of digits in amount fraction.

Referenced by frac_digits().

do_grouping()

```
template<typename _CharT, bool _Intl>
virtual string std::moneypunct< _CharT, _Intl >::do_grouping () const  [inline], [protected],
[virtual]
```

Return grouping specification.
Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

See also

grouping() for details.

Returns

String representing grouping specification.

Referenced by grouping().

do_neg_format()

```
template<typename _CharT, bool _Intl>
virtual pattern std::moneypunct< _CharT, _Intl >::do_neg_format () const  [inline], [protected],
[virtual]
```

Return pattern for money values.
This function returns a pattern describing the formatting of a negative valued money amount. This function is a hook for derived classes to change the value returned.

See also

neg_format() for details.

Returns

Pattern for money values.

Referenced by neg_format().

do_negative_sign()

```
template<typename _CharT, bool _Intl>
virtual string_type std::moneypunct< _CharT, _Intl >::do_negative_sign () const  [inline], [protected],
[virtual]
```

Return negative sign string.
This function returns a string_type to use as a sign for negative amounts. This function is a hook for derived classes to change the value returned.

See also

negative_sign() for details.

Returns

string_type representing a negative sign.

Referenced by negative_sign().

do_pos_format()

```
template<typename _CharT, bool _Intl>
virtual pattern std::moneypunct< _CharT, _Intl >::do_pos_format () const  [inline], [protected],
[virtual]
```

Return pattern for money values.
This function returns a pattern describing the formatting of a positive valued money amount. This function is a hook for derived classes to change the value returned.

See also

pos_format() for details.

Returns

Pattern for money values.

Referenced by pos_format().

do_positive_sign()

```
template<typename _CharT, bool _Intl>
virtual string_type std::moneypunct< _CharT, _Intl >::do_positive_sign () const  [inline], [protected],
[virtual]
```

Return positive sign string.
This function returns a string_type to use as a sign for positive amounts. This function is a hook for derived classes to change the value returned.

See also

positive_sign() for details.

Returns

string_type representing a positive sign.

Referenced by positive_sign().

do_thousands_sep()

```
template<typename _CharT, bool _Intl>
virtual char_type std::moneypunct< _CharT, _Intl >::do_thousands_sep () const  [inline], [protected],
[virtual]
```

Return thousands separator character.
Returns a char_type to use as a thousands separator. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a thousands separator.

Referenced by thousands_sep().

frac_digits()

```
template<typename _CharT, bool _Intl>
int std::moneypunct< _CharT, _Intl >::frac_digits () const  [inline]
```

Return number of digits in fraction.
This function returns the exact number of digits that make up the fractional part of a money amount. It does so by returning returning moneypunct<char_type>::do_frac_digits().
The fractional part of a money amount is optional. But if it is present, there must be frac_digits() digits.

Returns

Number of digits in amount fraction.

References do_frac_digits().

grouping()

```
template<typename _CharT, bool _Intl>
string std::moneypunct< _CharT, _Intl >::grouping () const  [inline]
```

Return grouping specification.
This function returns a string representing groupings for the integer part of an amount. Groupings indicate where thousands separators should be inserted.
Each char in the return string is interpret as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.
For example, if the grouping() returns `\003\002` and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was `32`, this would put more than 50 digits into the least significant group if the character set is ASCII.
The string is returned by calling moneypunct<char_type>::do_grouping().

Returns

string representing grouping specification.

References do_grouping().

neg_format()

```
template<typename _CharT, bool _Intl>
pattern std::moneypunct< _CharT, _Intl >::neg_format () const  [inline]
```

Return pattern for money values.
This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning returning moneypunct<char_type>::do_pos_format() or moneypunct<char_type>::do_neg_format().

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field. The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of curr_symbol() may be present. The sign field indicates that the value of positive_sign() or negative_sign() must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.
For example, for the US locale and pos_format() pattern {symbol,sign,value,none}, curr_symbol() == '$' positive_sign() == '+', and value 10.01, and options set to force the symbol, the corresponding string is `$+10.01`.

Returns

Pattern for money values.

References do_neg_format().

negative_sign()

```
template<typename _CharT, bool _Intl>
string_type std::moneypunct< _CharT, _Intl >::negative_sign () const  [inline]
```

Return negative sign string.
This function returns a string_type to use as a sign for negative amounts. It does so by returning returning moneypunct<char_type>::do_negative_sign().
If the return value contains more than one character, the first character appears in the position indicated by neg_format() and the remainder appear at the end of the formatted string.

Returns

string_type representing a negative sign.

References do_negative_sign().

pos_format()

```
template<typename _CharT, bool _Intl>
pattern std::moneypunct< _CharT, _Intl >::pos_format () const  [inline]
```

Return pattern for money values.
This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning returning moneypunct<char_type>::do_pos_format() or moneypunct<char_type>::do_neg_format().
The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field. The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of curr_symbol() may be present. The sign field indicates that the value of positive_sign() or negative_sign() must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.
For example, for the US locale and pos_format() pattern {symbol,sign,value,none}, curr_symbol() == '$' positive_sign() == '+', and value 10.01, and options set to force the symbol, the corresponding string is `$+10.01`.

Returns

Pattern for money values.

References do_pos_format().

positive_sign()

```
template<typename _CharT, bool _Intl>
string_type std::moneypunct< _CharT, _Intl >::positive_sign () const  [inline]
```

Return positive sign string.
This function returns a string_type to use as a sign for positive amounts. It does so by returning returning moneypunct<char_type>::do_positive_sign().
If the return value contains more than one character, the first character appears in the position indicated by pos_format() and the remainder appear at the end of the formatted string.

Returns

string_type representing a positive sign.

References do_positive_sign().

thousands_sep()

```
template<typename _CharT, bool _Intl>
char_type std::moneypunct< _CharT, _Intl >::thousands_sep () const  [inline]
```

Return thousands separator character.
This function returns a char_type to use as a thousands separator. It does so by returning returning moneypunct<char_type>::do_thousands_sep().

Returns

char_type representing a thousands separator.

References do_thousands_sep().

5.676.5 Member Data Documentation

id

```
template<typename _CharT, bool _Intl>
locale::id std::moneypunct< _CharT, _Intl >::id  [static]
```

Numpunct facet id.

intl

```
template<typename _CharT, bool _Intl>
const bool std::moneypunct< _CharT, _Intl >::intl  [static]
```

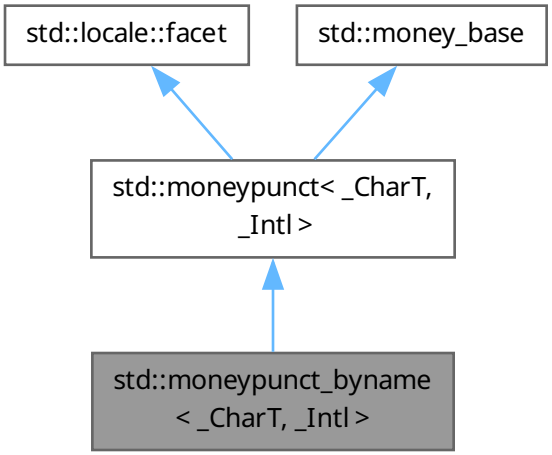
This value is provided by the standard, but no reason for its existence.
The documentation for this class was generated from the following file:

- locale_facets_nonio.h

5.677 std::moneypunct_byname< _CharT, _Intl > Class Template Reference

```
#include <locale_facets_nonio.h>
```

Inheritance diagram for std::moneypunct_byname< _CharT, _Intl >:



Public Types

- enum { **_S_minus** , **_S_zero** , **_S_end** }
- typedef __moneypunct_cache< _CharT, _Intl > **__cache_type**
- typedef _CharT **char_type**
- enum **part** {
  none , **space** , **symbol** , **sign** ,
  value }
- typedef basic_string< _CharT > **string_type**

Public Member Functions

- **moneypunct_byname** (const char *__s, size_t __refs=0)
- **moneypunct_byname** (const string &__s, size_t __refs=0)
- string_type curr_symbol () const
- char_type decimal_point () const
- int frac_digits () const
- string grouping () const
- string_type negative_sign () const
- string_type positive_sign () const
- char_type thousands_sep () const

- pattern pos_format () const
- pattern neg_format () const

Static Public Member Functions

- static pattern **_S_construct_pattern** (char __precedes, char __space, char __posn) throw ()

Static Public Attributes

- static const char * **_S_atoms**
- static const pattern **_S_default_pattern**
- static locale::id id
- static const bool **intl**

Protected Member Functions

- void **_M_initialize_moneypunct** (__c_locale __cloc=0, const char *__name=0)
- void **_M_initialize_moneypunct** (__c_locale, const char *)
- void **_M_initialize_moneypunct** (__c_locale, const char *)
- void **_M_initialize_moneypunct** (__c_locale, const char *)
- void **_M_initialize_moneypunct** (__c_locale, const char *)
- virtual string_type do_curr_symbol () const
- virtual char_type do_decimal_point () const
- virtual int do_frac_digits () const
- virtual string do_grouping () const
- virtual pattern do_neg_format () const
- virtual string_type do_negative_sign () const
- virtual pattern do_pos_format () const
- virtual string_type do_positive_sign () const
- virtual char_type do_thousands_sep () const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

5.677.1 Detailed Description

template<typename _CharT, bool _Intl>
class std::moneypunct_byname< _CharT, _Intl >

class moneypunct_byname [22.2.6.4].

5.677.2 Member Function Documentation

curr_symbol()

```
template<typename _CharT, bool _Intl>
string_type std::moneypunct< _CharT, _Intl >::curr_symbol () const  [inline], [inherited]
```

Return currency symbol string.
This function returns a string_type to use as a currency symbol. It does so by returning returning moneypunct<char↩
_type>::do_curr_symbol().

Returns

string_type representing a currency symbol.

References do_curr_symbol().

decimal_point()

```
template<typename _CharT, bool _Intl>
char_type std::moneypunct< _CharT, _Intl >::decimal_point () const  [inline], [inherited]
```

Return decimal point character.
This function returns a char_type to use as a decimal point. It does so by returning returning moneypunct<char_↩ type>::do_decimal_point().

Returns

char_type representing a decimal point.

References do_decimal_point().

do_curr_symbol()

```
template<typename _CharT, bool _Intl>
virtual string_type std::moneypunct< _CharT, _Intl >::do_curr_symbol () const  [inline], [protected],
[virtual], [inherited]
```

Return currency symbol string.
This function returns a string_type to use as a currency symbol. This function is a hook for derived classes to change the value returned.

See also

curr_symbol() for details.

Returns

string_type representing a currency symbol.

Referenced by curr_symbol().

do_decimal_point()

```
template<typename _CharT, bool _Intl>
virtual char_type std::moneypunct< _CharT, _Intl >::do_decimal_point () const  [inline], [protected],
[virtual], [inherited]
```

Return decimal point character.
Returns a char_type to use as a decimal point. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a decimal point.

Referenced by decimal_point().

do_frac_digits()

```
template<typename _CharT, bool _Intl>
virtual int std::moneypunct< _CharT, _Intl >::do_frac_digits () const  [inline], [protected],
[virtual], [inherited]
```

Return number of digits in fraction.
This function returns the exact number of digits that make up the fractional part of a money amount. This function is a hook for derived classes to change the value returned.

See also

frac_digits() for details.

Returns

Number of digits in amount fraction.

Referenced by frac_digits().

do_grouping()

```
template<typename _CharT, bool _Intl>
virtual string std::moneypunct< _CharT, _Intl >::do_grouping () const  [inline], [protected],
[virtual], [inherited]
```

Return grouping specification.
Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

See also

grouping() for details.

Returns

String representing grouping specification.

Referenced by grouping().

do_neg_format()

```
template<typename _CharT, bool _Intl>
virtual pattern std::moneypunct< _CharT, _Intl >::do_neg_format () const  [inline], [protected],
[virtual], [inherited]
```

Return pattern for money values.
This function returns a pattern describing the formatting of a negative valued money amount. This function is a hook for derived classes to change the value returned.

See also

neg_format() for details.

Returns

Pattern for money values.

Referenced by neg_format().

do_negative_sign()

```
template<typename _CharT, bool _Intl>
virtual string_type std::moneypunct< _CharT, _Intl >::do_negative_sign () const  [inline], [protected],
[virtual], [inherited]
```

Return negative sign string.
This function returns a string_type to use as a sign for negative amounts. This function is a hook for derived classes to change the value returned.

See also

negative_sign() for details.

Returns

string_type representing a negative sign.

Referenced by negative_sign().

do_pos_format()

```
template<typename _CharT, bool _Intl>
virtual pattern std::moneypunct< _CharT, _Intl >::do_pos_format () const  [inline], [protected],
[virtual], [inherited]
```

Return pattern for money values.

This function returns a pattern describing the formatting of a positive valued money amount. This function is a hook for derived classes to change the value returned.

See also

pos_format() for details.

Returns

Pattern for money values.

Referenced by pos_format().

do_positive_sign()

```
template<typename _CharT, bool _Intl>
virtual string_type std::moneypunct< _CharT, _Intl >::do_positive_sign () const  [inline], [protected],
[virtual], [inherited]
```

Return positive sign string.

This function returns a string_type to use as a sign for positive amounts. This function is a hook for derived classes to change the value returned.

See also

positive_sign() for details.

Returns

string_type representing a positive sign.

Referenced by positive_sign().

do_thousands_sep()

```
template<typename _CharT, bool _Intl>
virtual char_type std::moneypunct< _CharT, _Intl >::do_thousands_sep () const  [inline], [protected],
[virtual], [inherited]
```

Return thousands separator character.

Returns a char_type to use as a thousands separator. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a thousands separator.

Referenced by thousands_sep().

frac_digits()

```
template<typename _CharT, bool _Intl>
int std::moneypunct< _CharT, _Intl >::frac_digits () const  [inline], [inherited]
```

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. It does so by returning returning moneypunct<char_type>::do_frac_digits().

The fractional part of a money amount is optional. But if it is present, there must be frac_digits() digits.

Returns

Number of digits in amount fraction.

References do_frac_digits().

grouping()

```
template<typename _CharT, bool _Intl>
string std::moneypunct< _CharT, _Intl >::grouping () const  [inline], [inherited]
```

Return grouping specification.

This function returns a string representing groupings for the integer part of an amount. Groupings indicate where thousands separators should be inserted.

Each char in the return string is interpret as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the grouping() returns `\003\002` and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was `32`, this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling moneypunct<char_type>::do_grouping().

Returns

string representing grouping specification.

References do_grouping().

neg_format()

```
template<typename _CharT, bool _Intl>
pattern std::moneypunct< _CharT, _Intl >::neg_format () const  [inline], [inherited]
```

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning returning moneypunct<char_type>::do_pos_format() or moneypunct<char_type>::do_neg_format().

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of curr_symbol() may be present. The sign field indicates that the value of positive_sign() or negative_↩ sign() must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and pos_format() pattern {symbol,sign,value,none}, curr_symbol() == '$' positive_sign() == '+', and value 10.01, and options set to force the symbol, the corresponding string is `$+10.01`.

Returns

Pattern for money values.

References do_neg_format().

negative_sign()

```
template<typename _CharT, bool _Intl>
string_type std::moneypunct< _CharT, _Intl >::negative_sign () const  [inline], [inherited]
```

Return negative sign string.

This function returns a string_type to use as a sign for negative amounts. It does so by returning returning moneypunct<char_type>::do_negative_sign().

If the return value contains more than one character, the first character appears in the position indicated by neg_format() and the remainder appear at the end of the formatted string.

Returns

string_type representing a negative sign.

References do_negative_sign().

pos_format()

```
template<typename _CharT, bool _Intl>
pattern std::moneypunct< _CharT, _Intl >::pos_format () const  [inline], [inherited]
```

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning returning moneypunct<char_type>::do_pos_format() or moneypunct<char_type>::do_neg_format().

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of curr_symbol() may be present. The sign field indicates that the value of positive_sign() or negative_sign() must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and pos_format() pattern {symbol,sign,value,none}, curr_symbol() == '$' positive_sign() == '+', and value 10.01, and options set to force the symbol, the corresponding string is `$+10.01`.

Returns

Pattern for money values.

References do_pos_format().

positive_sign()

```
template<typename _CharT, bool _Intl>
string_type std::moneypunct< _CharT, _Intl >::positive_sign () const  [inline], [inherited]
```

Return positive sign string.

This function returns a string_type to use as a sign for positive amounts. It does so by returning returning moneypunct<char_type>::do_positive_sign().

If the return value contains more than one character, the first character appears in the position indicated by pos_format() and the remainder appear at the end of the formatted string.

Returns

string_type representing a positive sign.

References do_positive_sign().

thousands_sep()

```
template<typename _CharT, bool _Intl>
char_type std::moneypunct< _CharT, _Intl >::thousands_sep () const  [inline], [inherited]
```

Return thousands separator character.

This function returns a char_type to use as a thousands separator. It does so by returning returning moneypunct<char_type>::do_thousands_sep().

Returns

char_type representing a thousands separator.

References do_thousands_sep().

5.677.3 Member Data Documentation

id

```
template<typename _CharT, bool _Intl>
locale::id std::moneypunct< _CharT, _Intl >::id  [static], [inherited]
```

Numpunct facet id.

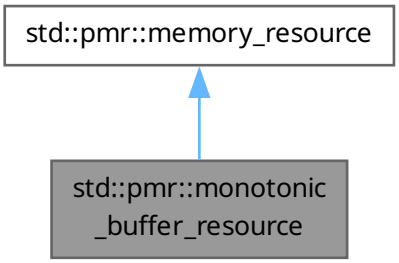
The documentation for this class was generated from the following file:

- locale_facets_nonio.h

5.678 std::pmr::monotonic_buffer_resource Class Reference

```
#include <memory_resource>
```

Inheritance diagram for std::pmr::monotonic_buffer_resource:



Public Member Functions

- **monotonic_buffer_resource** (const monotonic_buffer_resource &)=delete
- **monotonic_buffer_resource** (memory_resource *__upstream) noexcept
- **monotonic_buffer_resource** (size_t __initial_size) noexcept
- **monotonic_buffer_resource** (size_t __initial_size, memory_resource *__upstream) noexcept
- **monotonic_buffer_resource** (void *__buffer, size_t __buffer_size) noexcept
- **monotonic_buffer_resource** (void *__buffer, size_t __buffer_size, memory_resource *__upstream) noexcept
- void * **allocate** (size_t __bytes, size_t __alignment=_S_max_align)
- void **deallocate** (void *__p, size_t __bytes, size_t __alignment=_S_max_align)
- bool **is_equal** (const memory_resource &__other) const noexcept
- monotonic_buffer_resource & **operator=** (const monotonic_buffer_resource &)=delete
- void **release** () noexcept
- memory_resource * **upstream_resource** () const noexcept

Protected Member Functions

- void * do_allocate (size_t __bytes, size_t __alignment) override
- void do_deallocate (void *, size_t, size_t) override
- bool do_is_equal (const memory_resource &__other) const noexcept override

5.678.1 Detailed Description

A memory resource that allocates from a fixed-size buffer.
The main feature of a `pmr::monotonic_buffer_resource` is that its `do_deallocate` does nothing. This makes it very fast because there is no need to manage a free list, and every allocation simply returns a new block of memory, rather than searching for a suitably-sized free block. Because deallocating is a no-op, the amount of memory used by the resource only grows until `release()` (or the destructor) is called to return all memory to upstream.
A `monotonic_buffer_resource` can be initialized with a buffer that will be used to satisfy all allocation requests, until the buffer is full. After that a new buffer will be allocated from the upstream resource. By using a stack buffer and `pmr::null_memory_resource()` as the upstream you can get a memory resource that only uses the stack and never dynamically allocates.

Since

C++17

5.678.2 Member Function Documentation

do_allocate()

```
void * std::pmr::monotonic_buffer_resource::do_allocate (
            size_t __bytes,
            size_t __alignment)  [inline], [override], [protected], [virtual]
```

Implements std::pmr::memory_resource.

do_deallocate()

```
void std::pmr::monotonic_buffer_resource::do_deallocate (
            void * ,
            size_t ,
            size_t )  [inline], [override], [protected], [virtual]
```

Implements std::pmr::memory_resource.

do_is_equal()

```
bool std::pmr::monotonic_buffer_resource::do_is_equal (
            const memory_resource & __other) const  [inline], [override], [protected], [virtual],
[noexcept]
```

Implements std::pmr::memory_resource.
The documentation for this class was generated from the following file:

- memory_resource

5.679 std::move_iterator< _Iterator > Class Template Reference

`#include <stl_iterator.h>`

Public Types

- using **difference_type**
- using **iterator_concept**
- using **iterator_type**
- using **pointer**
- using **reference**
- using **value_type**

Public Member Functions

- template<typename _Iter>
  requires __convertible<_Iter>
  constexpr **move_iterator** (const move_iterator< _Iter > &__i)
- constexpr **move_iterator** (iterator_type __i)
- constexpr iterator_type **base** () &&
- constexpr const iterator_type & **base** () const &noexcept
- constexpr reference **operator*** () const
- constexpr move_iterator **operator+** (difference_type __n) const
- constexpr move_iterator & **operator++** ()
- constexpr move_iterator **operator++** (int)
- constexpr void **operator++** (int)
- constexpr move_iterator & **operator+=** (difference_type __n)
- constexpr move_iterator **operator-** (difference_type __n) const
- constexpr move_iterator & **operator--** ()
- constexpr move_iterator **operator--** (int)
- constexpr move_iterator & **operator-=** (difference_type __n)
- constexpr pointer **operator->** () const
- template<typename _Iter>
  requires __convertible<_Iter> && assignable_from<_Iterator&, const _Iter&>
  constexpr move_iterator & **operator=** (const move_iterator< _Iter > &__i)
- constexpr reference **operator[ ]** (difference_type __n) const

Friends

- constexpr iter_rvalue_reference_t< _Iterator > **iter_move** (const move_iterator &__i) noexcept(noexcept(ranges::iter_move(__i._M_current)))
- template<indirectly_swappable< _Iterator > _Iter2>
  constexpr void **iter_swap** (const move_iterator &__x, const move_iterator< _Iter2 > &__y) noexcept(noexcept(ranges::iter_swap(__x._M_current, __y._M_current)))
- template<sized_sentinel_for< _Iterator > _Sent>
  constexpr iter_difference_t< _Iterator > **operator-** (const move_iterator &__x, const move_sentinel< _Sent > &__y)
- template<sized_sentinel_for< _Iterator > _Sent>
  constexpr iter_difference_t< _Iterator > **operator-** (const move_sentinel< _Sent > &__x, const move_iterator &__y)
- template<sentinel_for< _Iterator > _Sent>
  constexpr bool **operator==** (const move_iterator &__x, const move_sentinel< _Sent > &__y)

5.679.1 Detailed Description

template<typename _Iterator>
class std::move_iterator< _Iterator >

An iterator adaptor that yields an rvalue reference.
Class template move_iterator is an iterator adapter with the same behavior as the underlying iterator except that its dereference operator implicitly converts the value returned by the underlying iterator's dereference operator to an rvalue reference. Some generic algorithms can be called with move iterators to replace copying with moving.
The documentation for this class was generated from the following file:

- bits/stl_iterator.h

5.680 std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> Class Template Reference

`#include <functional>`

Public Types

- using **result_type**

Public Member Functions

- move_only_function () noexcept
- template<typename _Fn, typename _Vt = decay_t<_Fn>>
  requires (!is_same_v<_Vt, move_only_function>) && (!__is_in_place_type_v<_Vt>) && __is_callable_from<_Vt>
  move_only_function (_Fn &&__f) noexcept(_S_nothrow_init< _Vt, _Fn >())
- template<typename _Tp, typename... _Args>
  requires is_constructible_v<_Tp, _Args...> && __is_callable_from<_Tp>
  move_only_function (in_place_type_t< _Tp >, _Args &&... __args) noexcept(_S_nothrow_init< _Tp, _Args... >())
- template<typename _Tp, typename _Up, typename... _Args>
  requires is_constructible_v<_Tp, initializer_list<_Up>&, _Args...> && __is_callable_from<_Tp>
  move_only_function (in_place_type_t< _Tp >, initializer_list< _Up > __il, _Args &&... __args) noexcept(_S_nothrow_init< _Tp, initializer_list< _Up > &, _Args... >())
- move_only_function (move_only_function &&__x) noexcept
- move_only_function (nullptr_t) noexcept
- operator bool () const noexcept
- _Res operator() (_ArgTypes... __args) _GLIBCXX_MOF_CV noexcept(_Noex)
- template<typename _Fn>
  requires is_constructible_v<move_only_function, _Fn>
  move_only_function & operator= (_Fn &&__f) noexcept(is_nothrow_constructible_v< move_only_function, _Fn >)
- move_only_function & operator= (move_only_function &&__x) noexcept
- move_only_function & operator= (nullptr_t) noexcept
- void swap (move_only_function &__x) noexcept

Friends

- template<typename _Func>
  auto & **__polyfunc::__base_of** (_Func &) noexcept
- template<typename _Func>
  auto & **__polyfunc::__invoker_of** (_Func &) noexcept
- template<typename _Dst, typename _Src>
  consteval bool **__polyfunc::__is_invoker_convertible** () noexcept
- bool operator== (const move_only_function &__x, nullptr_t) noexcept
- void swap (move_only_function &__x, move_only_function &__y) noexcept

5.680.1 Detailed Description

template<typename _Res, typename... _ArgTypes, bool _Noex>
class std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>

Polymorphic function wrapper.

Since

C++23

The `std::move_only_function` class template is a call wrapper similar to `std::function`, but does not require the stored target function to be copyable.
It also supports const-qualification, ref-qualification, and no-throw guarantees. The qualifications and exception-specification of the `move_only_function::operator()` member function are respected when invoking the target function.

5.680.2 Constructor & Destructor Documentation

move_only_function() [1/6]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::move_only_function
() [inline], [noexcept]
```

Creates an empty object.
References move_only_function().
Referenced by move_only_function(), move_only_function(), move_only_function(), move_only_function(), move_only_function(), move_only_function(), operator=(), operator=(), operator=(), operator==, swap(), and swap.

move_only_function() [2/6]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::move_only_function
(
            nullptr_t ) [inline], [noexcept]
```

Creates an empty object.
References move_only_function().

move_only_function() [3/6]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::move_only_function
(
            move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> && __x)
[inline], [noexcept]
```

Moves the target object, leaving the source empty.
References move_only_function().

move_only_function() [4/6]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<typename _Fn, typename _Vt = decay_t<_Fn>>
requires (!is_same_v<_Vt, move_only_function>) && (!__is_in_place_type_v<_Vt>) && __is_callable↩
_from<_Vt>
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::move_only_function
(
            _Fn && __f) [inline], [noexcept]
```

Stores a target object initialized from the argument.
References move_only_function(), and std::forward().

move_only_function() [5/6]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<typename _Tp, typename... _Args>
```

```
requires is_constructible_v<_Tp, _Args...> && __is_callable_from<_Tp>
std::move_only_function< _Res(_ArgTypes...)  _GLIBCXX_MOF_CV noexcept(_Noex)>::move_only_function
(
            in_place_type_t< _Tp > ,
            _Args &&...  __args)  [inline], [explicit], [noexcept]
```

Stores a target object initialized from the arguments.
References move_only_function(), and std::forward().

move_only_function() [6/6]

```
template<typename _Res, typename...  _ArgTypes, bool _Noex>
template<typename _Tp, typename _Up, typename...  _Args>
requires is_constructible_v<_Tp, initializer_list<_Up>&, _Args...> && __is_callable_from<_Tp>
std::move_only_function< _Res(_ArgTypes...)  _GLIBCXX_MOF_CV noexcept(_Noex)>::move_only_function
(
            in_place_type_t< _Tp > ,
            initializer_list< _Up > __il,
            _Args &&...  __args)  [inline], [explicit], [noexcept]
```

Stores a target object initialized from the arguments.
References move_only_function(), and std::forward().

5.680.3 Member Function Documentation

operator bool()

```
template<typename _Res, typename...  _ArgTypes, bool _Noex>
std::move_only_function< _Res(_ArgTypes...)  _GLIBCXX_MOF_CV noexcept(_Noex)>::operator bool ()
const  [inline], [explicit], [noexcept]
```

True if a target object is present, false otherwise.

operator()()

```
template<typename _Res, typename...  _ArgTypes, bool _Noex>
_Res std::move_only_function< _Res(_ArgTypes...)  _GLIBCXX_MOF_CV noexcept(_Noex)>::operator() (
            _ArgTypes...  __args)  [inline], [noexcept]
```

Invoke the target object.
The target object will be invoked using the supplied arguments, and as an lvalue or rvalue, and as const or non-const, as dictated by the template arguments of the `move_only_function` specialization.

Precondition

Must not be empty.

References std::forward(), and operator()().
Referenced by operator()().

operator=() [1/3]

```
template<typename _Res, typename...  _ArgTypes, bool _Noex>
template<typename _Fn>
requires is_constructible_v<move_only_function, _Fn>
move_only_function & std::move_only_function< _Res(_ArgTypes...)  _GLIBCXX_MOF_CV noexcept(_↩
Noex)>::operator= (
            _Fn && __f)  [inline], [noexcept]
```

Stores a new target object, initialized from the argument.
References move_only_function(), std::forward(), and operator=().

operator=() [2/3]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
move_only_function & std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_↩
Noex)>::operator= (
            move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> && __x)
[inline], [noexcept]
```

Stores a new target object, leaving `x` empty.
References move_only_function(), std::addressof(), and operator=().
Referenced by operator=(), operator=(), and operator=().

operator=() [3/3]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
move_only_function & std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_↩
Noex)>::operator= (
            nullptr_t ) [inline], [noexcept]
```

Destroys the target object (if any).
References move_only_function(), and operator=().

swap()

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
void std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::swap (
            move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> & __x)
[inline], [noexcept]
```

Exchange the target objects (if any).
References move_only_function(), and swap().
Referenced by swap(), and swap.

5.680.4 Friends And Related Symbol Documentation

operator==

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
bool operator== (
            const move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> & _↩
_x,
            nullptr_t ) [friend]
```

Check for emptiness by comparing with `nullptr`.
References move_only_function(), and operator==.
Referenced by operator==.

swap

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
void swap (
            move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> & __x,
            move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> & __y)
[friend]
```

Exchange the target objects (if any).
References move_only_function(), and swap().
The documentation for this class was generated from the following file:

- mofunc_impl.h

5.681 std::move_sentinel< _Sent > Class Template Reference

`#include <stl_iterator.h>`

Public Member Functions

- constexpr **move_sentinel** (_Sent __s) noexcept(is_nothrow_move_constructible_v< _Sent >)
- template<typename _S2>
  requires convertible_to<const _S2&, _Sent>
  constexpr **move_sentinel** (const move_sentinel< _S2 > &__s) noexcept(is_nothrow_constructible_v< _Sent, const _S2 & >)
- constexpr _Sent **base** () const noexcept(is_nothrow_copy_constructible_v< _Sent >)
- template<typename _S2>
  requires assignable_from<_Sent&, const _S2&>
  constexpr move_sentinel & **operator=** (const move_sentinel< _S2 > &__s) noexcept(is_nothrow_assignable_v< _Sent, const _S2 & >)

5.681.1 Detailed Description

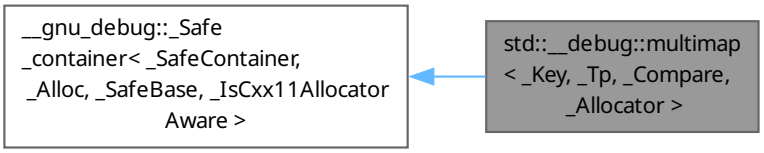
template<semiregular _Sent>
class std::move_sentinel< _Sent >

A sentinel adaptor for use with std::move_iterator.
The documentation for this class was generated from the following file:

- bits/stl_iterator.h

5.682 std::__debug::multimap< _Key, _Tp, _Compare, _Allocator > Class Template Reference

`#include <multimap.h>`
Inheritance diagram for std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >:



Public Types

- typedef _Allocator **allocator_type**
- typedef __gnu_debug::_Safe_iterator< _Base_const_iterator, multimap > **const_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**
- typedef std::reverse_iterator< const_iterator > **const_reverse_iterator**
- typedef _Base::difference_type **difference_type**
- typedef __gnu_debug::_Safe_iterator< _Base_iterator, multimap > **iterator**
- typedef _Compare **key_compare**

- typedef _Key **key_type**
- typedef _Tp **mapped_type**
- using **node_type**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef std::reverse_iterator< iterator > **reverse_iterator**
- typedef _Base::size_type **size_type**
- typedef std::pair< const _Key, _Tp > **value_type**

Public Member Functions

- **multimap** (_Base_ref __x)
- template<typename _InputIterator>
  multimap (_InputIterator __first, _InputIterator __last, const _Compare &__comp=_Compare(), const _Allocator &__a=_Allocator())
- template<typename _InputIterator>
  multimap (_InputIterator __first, _InputIterator __last, const allocator_type &__a)
- **multimap** (const _Compare &__comp, const _Allocator &__a=_Allocator())
- **multimap** (const allocator_type &__a)
- **multimap** (const multimap &)=default
- **multimap** (const multimap &__m, const __type_identity_t< allocator_type > &__a)
- **multimap** (initializer_list< value_type > __l, const _Compare &__c=_Compare(), const allocator_type &__↩ a=allocator_type())
- **multimap** (initializer_list< value_type > __l, const allocator_type &__a)
- **multimap** (multimap &&)=default
- **multimap** (multimap &&__m, const __type_identity_t< allocator_type > &__a) noexcept(noexcept(_Base(std::move(↩ __m), __a)))
- **__attribute** ((__abi_tag__("cxx11"))) iterator erase(iterator __position)
- const _Base & **_M_base** () const noexcept
- _Base & **_M_base** () noexcept
- const_iterator **begin** () const noexcept
- iterator **begin** () noexcept
- const_iterator **cbegin** () const noexcept
- const_iterator **cend** () const noexcept
- void **clear** () noexcept
- const_reverse_iterator **crbegin** () const noexcept
- const_reverse_iterator **crend** () const noexcept
- template<typename... _Args>
  iterator **emplace** (_Args &&... __args)
- template<typename... _Args>
  iterator **emplace_hint** (const_iterator __pos, _Args &&... __args)
- const_iterator **end** () const noexcept
- iterator **end** () noexcept
- template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>
  std::pair< iterator, iterator > **equal_range** (const _Kt &__x)
- template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>
  std::pair< const_iterator, const_iterator > **equal_range** (const _Kt &__x) const
- std::pair< iterator, iterator > **equal_range** (const key_type &__x)
- std::pair< const_iterator, const_iterator > **equal_range** (const key_type &__x) const
- _Base_iterator **erase** (_Base_const_iterator __position)
- size_type **erase** (const key_type &__x)
- iterator **erase** (const_iterator __first, const_iterator __last)

- iterator **erase** (const_iterator __position)
- node_type **extract** (const key_type &__key)
- node_type **extract** (const_iterator __position)
- template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>
  iterator **find** (const _Kt &__x)
- template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>
  const_iterator **find** (const _Kt &__x) const
- iterator **find** (const key_type &__x)
- const_iterator **find** (const key_type &__x) const
- template<typename _InputIterator>
  void **insert** (_InputIterator __first, _InputIterator __last)
- template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
  iterator **insert** (_Pair &&__x)
- iterator **insert** (const value_type &__x)
- iterator **insert** (const_iterator __hint, node_type &&__nh)
- template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
  iterator **insert** (const_iterator __position, _Pair &&__x)
- iterator **insert** (const_iterator __position, const value_type &__x)
- iterator **insert** (const_iterator __position, value_type &&__x)
- iterator **insert** (node_type &&__nh)
- void **insert** (std::initializer_list< value_type > __list)
- iterator **insert** (value_type &&__x)
- template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>
  iterator **lower_bound** (const _Kt &__x)
- template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>
  const_iterator **lower_bound** (const _Kt &__x) const
- iterator **lower_bound** (const key_type &__x)
- const_iterator **lower_bound** (const key_type &__x) const
- multimap & **operator=** (const multimap &)=default
- multimap & **operator=** (initializer_list< value_type > __l)
- multimap & **operator=** (multimap &&)=default
- const_reverse_iterator **rbegin** () const noexcept
- reverse_iterator **rbegin** () noexcept
- const_reverse_iterator **rend** () const noexcept
- reverse_iterator **rend** () noexcept
- void **swap** (multimap &__x) noexcept(/*conditional */)
- template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>
  iterator **upper_bound** (const _Kt &__x)
- template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>
  const_iterator **upper_bound** (const _Kt &__x) const
- iterator **upper_bound** (const key_type &__x)
- const_iterator **upper_bound** (const key_type &__x) const

Protected Member Functions

- constexpr void **_M_swap** (const _Safe_container &__x) const noexcept

Friends

- template<typename _ItT, typename _SeqT, typename _CatT>
  class **::__gnu_debug::_Safe_iterator**

5.682.1 Detailed Description

template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<std::pair<const _Key, _Tp> >>
class std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >

Class std::multimap with safety/checking/debug instrumentation.
The documentation for this class was generated from the following file:

- multimap.h

5.683 std::multimap< _Key, _Tp, _Compare, _Alloc > Class Template Reference

`#include <map>`

Public Types

- typedef _Alloc **allocator_type**
- typedef _Rep_type::const_iterator **const_iterator**
- typedef _Alloc_traits::const_pointer **const_pointer**
- typedef _Alloc_traits::const_reference **const_reference**
- typedef _Rep_type::const_reverse_iterator **const_reverse_iterator**
- typedef _Rep_type::difference_type **difference_type**
- typedef _Rep_type::iterator **iterator**
- typedef _Compare **key_compare**
- typedef _Key **key_type**
- typedef _Tp **mapped_type**
- typedef _Alloc_traits::pointer **pointer**
- typedef _Alloc_traits::reference **reference**
- typedef _Rep_type::reverse_iterator **reverse_iterator**
- typedef _Rep_type::size_type **size_type**
- typedef std::pair< const _Key, _Tp > **value_type**

Public Member Functions

- multimap ()=default
- template<typename _InputIterator>
  multimap (_InputIterator __first, _InputIterator __last)
- template<typename _InputIterator>
  multimap (_InputIterator __first, _InputIterator __last, const _Compare &__comp, const allocator_type &__a=allocator_type())
- template<typename _InputIterator>
  multimap (_InputIterator __first, _InputIterator __last, const allocator_type &__a)
- multimap (const _Compare &__comp, const allocator_type &__a=allocator_type())
- multimap (const allocator_type &__a)
- multimap (const multimap &)=default
- multimap (const multimap &__m, const __type_identity_t< allocator_type > &__a)
- multimap (initializer_list< value_type > __l, const _Compare &__comp=_Compare(), const allocator_type &__a=allocator_type())
- multimap (initializer_list< value_type > __l, const allocator_type &__a)
- multimap (multimap &&)=default
- multimap (multimap &&__m, const __type_identity_t< allocator_type > &__a) noexcept(is_nothrow_copy_constructible< _Compare >::value &&_Alloc_traits::_S_always_equal())
- ~multimap ()=default

- const_iterator begin () const noexcept
- iterator begin () noexcept
- const_iterator cbegin () const noexcept
- const_iterator cend () const noexcept
- void clear () noexcept
- const_reverse_iterator crbegin () const noexcept
- const_reverse_iterator crend () const noexcept
- template<typename... _Args>
  iterator emplace (_Args &&... __args)
- template<typename... _Args>
  iterator emplace_hint (const_iterator __pos, _Args &&... __args)
- bool empty () const noexcept
- const_iterator end () const noexcept
- iterator end () noexcept
- size_type erase (const key_type &__x)
- iterator erase (const_iterator __first, const_iterator __last)
- allocator_type get_allocator () const noexcept
- template<typename _InputIterator>
  void insert (_InputIterator __first, _InputIterator __last)
- void insert (initializer_list< value_type > __l)
- key_compare key_comp () const
- size_type max_size () const noexcept
- multimap & operator= (const multimap &)=default
- multimap & operator= (initializer_list< value_type > __l)
- multimap & operator= (multimap &&)=default
- const_reverse_iterator rbegin () const noexcept
- reverse_iterator rbegin () noexcept
- const_reverse_iterator rend () const noexcept
- reverse_iterator rend () noexcept
- size_type size () const noexcept
- void swap (multimap &__x) noexcept(/*conditional */)
- value_compare value_comp () const

- iterator insert (const value_type &__x)
- iterator insert (value_type &&__x)
- template<typename _Pair>
  __enable_if_t< is_constructible< value_type, _Pair >::value, iterator > insert (_Pair &&__x)

- iterator insert (const_iterator __position, const value_type &__x)
- iterator insert (const_iterator __position, value_type &&__x)
- template<typename _Pair>
  __enable_if_t< is_constructible< value_type, _Pair && >::value, iterator > insert (const_iterator __position, _↩
  Pair &&__x)

- iterator erase (const_iterator __position)
- _GLIBCXX_ABI_TAG_CXX11 iterator erase (iterator __position)

- iterator find (const key_type &__x)
- template<typename _Kt>
  auto find (const _Kt &__x) -> decltype(_M_t._M_find_tr(__x))

- const_iterator find (const key_type &__x) const
- template<typename _Kt>
  auto find (const _Kt &__x) const -> decltype(_M_t._M_find_tr(__x))

- size_type count (const key_type &__x) const
- template<typename _Kt>
  auto count (const _Kt &__x) const -> decltype(_M_t._M_count_tr(__x))

- bool contains (const key_type &__x) const
- template<typename _Kt>
  auto contains (const _Kt &__x) const -> decltype(_M_t._M_find_tr(__x), void(), true)

- iterator lower_bound (const key_type &__x)
- template<typename _Kt>
  auto lower_bound (const _Kt &__x) -> decltype(iterator(_M_t._M_lower_bound_tr(__x)))

- const_iterator lower_bound (const key_type &__x) const
- template<typename _Kt>
  auto lower_bound (const _Kt &__x) const -> decltype(const_iterator(_M_t._M_lower_bound_tr(__x)))

- iterator upper_bound (const key_type &__x)
- template<typename _Kt>
  auto upper_bound (const _Kt &__x) -> decltype(iterator(_M_t._M_upper_bound_tr(__x)))

- const_iterator upper_bound (const key_type &__x) const
- template<typename _Kt>
  auto upper_bound (const _Kt &__x) const -> decltype(const_iterator(_M_t._M_upper_bound_tr(__x)))

- std::pair< iterator, iterator > equal_range (const key_type &__x)
- template<typename _Kt>
  auto equal_range (const _Kt &__x) -> decltype(pair< iterator, iterator >(_M_t._M_equal_range_tr(__x)))

- std::pair< const_iterator, const_iterator > equal_range (const key_type &__x) const
- template<typename _Kt>
  auto equal_range (const _Kt &__x) const -> decltype(pair< const_iterator, const_iterator >(_M_t._M_equal_↩
  range_tr(__x)))

Friends

- template<typename _K1, typename _T1, typename _C1, typename _A1>
  bool **operator**< (const multimap< _K1, _T1, _C1, _A1 > &, const multimap< _K1, _T1, _C1, _A1 > &)
- template<typename _K1, typename _T1, typename _C1, typename _A1>
  bool **operator==** (const multimap< _K1, _T1, _C1, _A1 > &, const multimap< _K1, _T1, _C1, _A1 > &)

5.683.1 Detailed Description

template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> > >
class std::multimap< _Key, _Tp, _Compare, _Alloc >

A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.

Since

C++98

Template Parameters

_Key	Type of key objects.
_Tp	Type of mapped objects.
_Compare	Comparison function object type, defaults to less<_Key>.
_Alloc	Allocator type, defaults to allocator<pair<const _Key, _Tp>.

Meets the requirements of a container, a reversible container, and an associative container (using equivalent keys). For a multimap<Key,T> the key_type is Key, the mapped_type is T, and the value_type is std::pair<const Key,T>.
Multimaps support bidirectional iterators.
The private tree data is declared exactly the same way for map and multimap; the distinction is made entirely in how the tree functions are called (*_unique versus *_equal, same as the standard).

5.683.2 Constructor & Destructor Documentation

multimap() [1/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap ()  [default]
```

Default constructor creates no elements.

multimap() [2/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
            const _Compare & __comp,
            const allocator_type & __a = allocator_type())  [inline], [explicit]
```

Creates a multimap with no elements.

Parameters

__comp	A comparison object.
__a	An allocator object.

multimap() [3/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
            const multimap< _Key, _Tp, _Compare, _Alloc > & )  [default]
```

Multimap copy constructor.
Whether the allocator is copied depends on the allocator traits.

multimap() [4/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
            multimap< _Key, _Tp, _Compare, _Alloc > && )  [default]
```

Multimap move constructor.
The newly-created multimap contains the exact contents of the moved instance. The moved instance is a valid, but unspecified multimap.

multimap() [5/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
            initializer_list< value_type > __l,
            const _Compare & __comp = _Compare(),
            const allocator_type & __a = allocator_type())  [inline]
```

Builds a multimap from an initializer_list.

Parameters

__l	An initializer_list.
__comp	A comparison functor.
__a	An allocator object.

Create a multimap consisting of copies of the elements from the initializer_list. This is linear in N if the list is already sorted, and NlogN otherwise (where N is *__l.size()*).

multimap() [6/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
            const allocator_type & __a)  [inline], [explicit]
```

Allocator-extended default constructor.

multimap() [7/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
            const multimap< _Key, _Tp, _Compare, _Alloc > & __m,
            const __type_identity_t< allocator_type > & __a)  [inline]
```

Allocator-extended copy constructor.

multimap() [8/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
            multimap< _Key, _Tp, _Compare, _Alloc > && __m,
            const __type_identity_t< allocator_type > & __a)  [inline], [noexcept]
```

Allocator-extended move constructor.

multimap() [9/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
            initializer_list< value_type > __l,
            const allocator_type & __a)  [inline]
```

Allocator-extended initialier-list constructor.

multimap() [10/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _InputIterator>
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
            _InputIterator __first,
            _InputIterator __last,
            const allocator_type & __a)  [inline]
```

Allocator-extended range constructor.

multimap() [11/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _InputIterator>
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
            _InputIterator __first,
            _InputIterator __last)  [inline]
```

Builds a multimap from a range.

Parameters

__first	An input iterator.
__last	An input iterator.

Create a multimap consisting of copies of the elements from [__first,__last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(__first,__last)).

multimap() [12/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _InputIterator>
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
            _InputIterator __first,
            _InputIterator __last,
```

```
        const _Compare & __comp,
        const allocator_type & __a = allocator_type())  [inline]
```

Builds a multimap from a range.

Parameters

__first	An input iterator.
__last	An input iterator.
__comp	A comparison functor.
__a	An allocator object.

Create a multimap consisting of copies of the elements from [__first,__last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(__first,__last)).

~multimap()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap< _Key, _Tp, _Compare, _Alloc >::~multimap ()  [default]
```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

5.683.3 Member Function Documentation

begin() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::begin () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first pair in the multimap. Iteration is done in ascending order according to the keys.

begin() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::begin ()  [inline], [noexcept]
```

Returns a read/write iterator that points to the first pair in the multimap. Iteration is done in ascending order according to the keys.

cbegin()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::cbegin () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first pair in the multimap. Iteration is done in ascending order according to the keys.

cend()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::cend () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last pair in the multimap. Iteration is done in ascending order according to the keys.

clear()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
void std::multimap< _Key, _Tp, _Compare, _Alloc >::clear ()  [inline], [noexcept]
```

Erases all elements in a multimap. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

contains() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt>
auto std::multimap< _Key, _Tp, _Compare, _Alloc >::contains (
            const _Kt & __x) const -> decltype(_M_t._M_find_tr(__x), void(), true)   [inline]
```

Finds whether an element with the given key exists.

Parameters

_↩ _x	Key of (key, value) pairs to be located.

Returns

True if there is any element with the specified key.

contains() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
bool std::multimap< _Key, _Tp, _Compare, _Alloc >::contains (
            const key_type & __x) const  [inline]
```

Finds whether an element with the given key exists.

Parameters

_↩ _x	Key of (key, value) pairs to be located.

Returns

True if there is any element with the specified key.

count() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt>
auto std::multimap< _Key, _Tp, _Compare, _Alloc >::count (
            const _Kt & __x) const -> decltype(_M_t._M_count_tr(__x))   [inline]
```

Finds the number of elements with given key.

Parameters

_↩ _x	Key of (key, value) pairs to be located.

Returns

Number of elements with specified key.

count() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::multimap< _Key, _Tp, _Compare, _Alloc >::count (
            const key_type & __x) const  [inline]
```

Finds the number of elements with given key.

Parameters

_↩ _x	Key of (key, value) pairs to be located.

Returns

Number of elements with specified key.

crbegin()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::crbegin () const  [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the multimap. Iteration is done in descending order according to the keys.

crend()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::crend () const  [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first pair in the multimap. Iteration is done in descending order according to the keys.

emplace()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename... _Args>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::emplace (
            _Args &&... __args)  [inline]
```

Build and insert a std::pair into the multimap.

Parameters

__args	Arguments used to generate a new pair instance (see std::piecewise_contruct for passing arguments to each part of the pair constructor).

Returns

An iterator that points to the inserted (key,value) pair.

This function builds and inserts a (key, value) pair into the multimap. Contrary to a std::map the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted.
Insertion requires logarithmic time.

emplace_hint()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename... _Args>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::emplace_hint (
            const_iterator __pos,
            _Args &&... __args)  [inline]
```

Builds and inserts a std::pair into the multimap.

Parameters

__pos	An iterator that serves as a hint as to where the pair should be inserted.
__args	Arguments used to generate a new pair instance (see std::piecewise_contruct for passing arguments to each part of the pair constructor).

Returns

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a std::map the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
For more on *hinting*, see: https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints
Insertion requires logarithmic time (if the hint is not taken).

empty()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
bool std::multimap< _Key, _Tp, _Compare, _Alloc >::empty () const  [inline], [nodiscard], [noexcept]
```

Returns true if the multimap is empty.

end() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::end () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last pair in the multimap. Iteration is done in ascending order according to the keys.

end() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::end ()  [inline], [noexcept]
```

Returns a read/write iterator that points one past the last pair in the multimap. Iteration is done in ascending order according to the keys.

equal_range() [1/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt>
auto std::multimap< _Key, _Tp, _Compare, _Alloc >::equal_range (
            const _Kt & __x) -> decltype(pair<iterator, iterator>(_M_t._M_equal_range_tr(__x)))
[inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key of (key, value) pairs to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
               c.upper_bound(val))
```

(but is faster than making the calls separately).

equal_range() [2/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt>
auto std::multimap< _Key, _Tp, _Compare, _Alloc >::equal_range (
            const _Kt & __x) const -> decltype(pair<const_iterator, const_iterator>( _M_t._M_↩
equal_range_tr(__x)))   [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key of (key, value) pairs to be located.

Returns

Pair of read-only (constant) iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
               c.upper_bound(val))
```

(but is faster than making the calls separately).

equal_range() [3/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::pair< iterator, iterator > std::multimap< _Key, _Tp, _Compare, _Alloc >::equal_range (
            const key_type & __x)  [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key of (key, value) pairs to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
               c.upper_bound(val))
```

(but is faster than making the calls separately).

equal_range() [4/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::pair< const_iterator, const_iterator > std::multimap< _Key, _Tp, _Compare, _Alloc >::equal↩
_range (
            const key_type & __x) const  [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key of (key, value) pairs to be located.

Returns

Pair of read-only (constant) iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
               c.upper_bound(val))
```

(but is faster than making the calls separately).

erase() [1/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::multimap< _Key, _Tp, _Compare, _Alloc >::erase (
            const key_type & __x)  [inline]
```

Erases elements according to the provided key.

Parameters

_↩ _*x*	Key of element to be erased.

Returns

The number of elements erased.

This function erases all elements located by the given key from a multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [2/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::erase (
            const_iterator __first,
            const_iterator __last)  [inline]
```

Erases a [first,last) range of elements from a multimap.

Parameters

__first	Iterator pointing to the start of the range to be erased.
__last	Iterator pointing to the end of the range to be erased .

Returns

The iterator *__last*.

This function erases a sequence of elements from a multimap. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [3/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::erase (
            const_iterator __position)  [inline]
```

Erases an element from a multimap.

Parameters

__position	An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from a multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [4/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
_GLIBCXX_ABI_TAG_CXX11 iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::erase (
            iterator __position)  [inline]
```

Erases an element from a multimap.

Parameters

__position	An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from a multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

find() [1/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt>
auto std::multimap< _Key, _Tp, _Compare, _Alloc >::find (
            const _Kt & __x) -> decltype(_M_t._M_find_tr(__x))   [inline]
```

Tries to locate an element in a multimap.

Parameters

_x	Key of (key, value) pair to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end (`end()`) iterator.

find() [2/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt>
auto std::multimap< _Key, _Tp, _Compare, _Alloc >::find (
            const _Kt & __x) const -> decltype(_M_t._M_find_tr(__x))   [inline]
```

Tries to locate an element in a multimap.

Parameters

_x	Key of (key, value) pair to be located.

Returns

Read-only (constant) iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns a constant iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end (`end()`) iterator.

find() [3/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::find (
            const key_type & __x)  [inline]
```

Tries to locate an element in a multimap.

Parameters

_↩ _x	Key of (key, value) pair to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end (`end()`) iterator.

find() [4/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::find (
            const key_type & __x) const  [inline]
```

Tries to locate an element in a multimap.

Parameters

_↩ _x	Key of (key, value) pair to be located.

Returns

Read-only (constant) iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns a constant iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end (`end()`) iterator.

get_allocator()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
allocator_type std::multimap< _Key, _Tp, _Compare, _Alloc >::get_allocator () const  [inline],
[noexcept]
```

Get a copy of the memory allocation object.

insert() [1/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _InputIterator>
void std::multimap< _Key, _Tp, _Compare, _Alloc >::insert (
            _InputIterator __first,
            _InputIterator __last)  [inline]
```

A template function that attempts to insert a range of elements.

Parameters

__first	Iterator pointing to the start of the range to be inserted.
__last	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

insert() [2/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Pair>
__enable_if_t< is_constructible< value_type, _Pair >::value, iterator > std::multimap< _Key, _↩
Tp, _Compare, _Alloc >::insert (
            _Pair && __x)  [inline]
```

Inserts a std::pair into the multimap.

Parameters

_↩ _x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a std::map the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted.
Insertion requires logarithmic time.

insert() [3/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::insert (
            const value_type & __x)  [inline]
```

Inserts a std::pair into the multimap.

Parameters

_↩ _x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a std::map the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted.
Insertion requires logarithmic time.
Referenced by std::multimap< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert().

insert() [4/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Pair>
__enable_if_t< is_constructible< value_type, _Pair && >::value, iterator > std::multimap< _Key,
_Tp, _Compare, _Alloc >::insert (
            const_iterator __position,
            _Pair && __x)  [inline]
```

Inserts a std::pair into the multimap.

Parameters

__position	An iterator that serves as a hint as to where the pair should be inserted.
__x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a std::map the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
For more on *hinting*, see: https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints
Insertion requires logarithmic time (if the hint is not taken).

insert() [5/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::insert (
            const_iterator __position,
            const value_type & __x)  [inline]
```

Inserts a std::pair into the multimap.

Parameters

__position	An iterator that serves as a hint as to where the pair should be inserted.
__x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a std::map the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
For more on *hinting*, see: https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints
Insertion requires logarithmic time (if the hint is not taken).

insert() [6/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::insert (
            const_iterator __position,
            value_type && __x)  [inline]
```

Inserts a std::pair into the multimap.

Parameters

__position	An iterator that serves as a hint as to where the pair should be inserted.
__x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a std::map the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
For more on *hinting*, see: https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints
Insertion requires logarithmic time (if the hint is not taken).

insert() [7/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
void std::multimap< _Key, _Tp, _Compare, _Alloc >::insert (
            initializer_list< value_type > __l)  [inline]
```

Attempts to insert a list of std::pairs into the multimap.

Parameters

__l	A std::initializer_list<value_type> of pairs to be inserted.

Complexity similar to that of the range constructor.

insert() [8/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::insert (
            value_type && __x)  [inline]
```

Inserts a std::pair into the multimap.

Parameters

_↩ _x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a std::map the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted.
Insertion requires logarithmic time.

key_comp()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
key_compare std::multimap< _Key, _Tp, _Compare, _Alloc >::key_comp () const  [inline]
```

Returns the key comparison object out of which the multimap was constructed.

lower_bound() [1/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt>
auto std::multimap< _Key, _Tp, _Compare, _Alloc >::lower_bound (
            const _Kt & __x) -> decltype(iterator(_M_t._M_lower_bound_tr(__x)))   [inline]
```

Finds the beginning of a subsequence matching given key.

Parameters

_↩ _x	Key of (key, value) pair to be located.

Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

lower_bound() [2/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt>
```

```
auto std::multimap< _Key, _Tp, _Compare, _Alloc >::lower_bound (
            const _Kt & __x) const -> decltype(const_iterator(_M_t._M_lower_bound_tr(__x)))
[inline]
```

Finds the beginning of a subsequence matching given key.

Parameters

_↩ _x	Key of (key, value) pair to be located.

Returns

Read-only (constant) iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful the iterator will point to the next greatest element or, if no such greater element exists, to end().

lower_bound() [3/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::lower_bound (
            const key_type & __x)  [inline]
```

Finds the beginning of a subsequence matching given key.

Parameters

_↩ _x	Key of (key, value) pair to be located.

Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

lower_bound() [4/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::lower_bound (
            const key_type & __x) const  [inline]
```

Finds the beginning of a subsequence matching given key.

Parameters

_↩ _x	Key of (key, value) pair to be located.

Returns

Read-only (constant) iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful the iterator will point to the next greatest element or, if no such greater element exists, to end().

max_size()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::multimap< _Key, _Tp, _Compare, _Alloc >::max_size () const  [inline], [noexcept]
```

Returns the maximum size of the multimap.

operator=() [1/3]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
multimap & std::multimap< _Key, _Tp, _Compare, _Alloc >::operator= (
            const multimap< _Key, _Tp, _Compare, _Alloc > & )  [default]
```

Multimap assignment operator.
Whether the allocator is copied depends on the allocator traits.

operator=() [2/3]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
multimap & std::multimap< _Key, _Tp, _Compare, _Alloc >::operator= (
            initializer_list< value_type > __l)  [inline]
```

Multimap list assignment operator.

Parameters

↩ _↩ ↩ _↩ *l*	An initializer_list.

This function fills a multimap with copies of the elements in the initializer list *__l*.
Note that the assignment completely changes the multimap and that the resulting multimap's size is the same as the number of elements assigned.

operator=() [3/3]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
multimap & std::multimap< _Key, _Tp, _Compare, _Alloc >::operator= (
            multimap< _Key, _Tp, _Compare, _Alloc > && )  [default]
```

Move assignment operator.

rbegin() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::rbegin () const  [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the multimap. Iteration is done in descending order according to the keys.

rbegin() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::rbegin ()  [inline], [noexcept]
```

Returns a read/write reverse iterator that points to the last pair in the multimap. Iteration is done in descending order according to the keys.

rend() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::rend () const  [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first pair in the multimap. Iteration is done in descending order according to the keys.

rend() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::rend ()  [inline], [noexcept]
```

Returns a read/write reverse iterator that points to one before the first pair in the multimap. Iteration is done in descending order according to the keys.

size()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::multimap< _Key, _Tp, _Compare, _Alloc >::size () const  [inline], [noexcept]
```

Returns the size of the multimap.

swap()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
void std::multimap< _Key, _Tp, _Compare, _Alloc >::swap (
            multimap< _Key, _Tp, _Compare, _Alloc > & __x)  [inline], [noexcept]
```

Swaps data with another multimap.

Parameters

_↩ _x	A multimap of the same element and allocator types.

This exchanges the elements between two multimaps in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global std::swap() function is specialized such that std::swap(m1,m2) will feed to this function.
Whether the allocators are swapped depends on the allocator traits.
Referenced by std::swap().

upper_bound() [1/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
template<typename _Kt>
auto std::multimap< _Key, _Tp, _Compare, _Alloc >::upper_bound (
            const _Kt & __x) -> decltype(iterator(_M_t._M_upper_bound_tr(__x)))   [inline]
```

Finds the end of a subsequence matching given key.

Parameters

_↩ _x	Key of (key, value) pair to be located.

Returns

Iterator pointing to the first element greater than key, or end().

upper_bound() [2/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt>
auto std::multimap< _Key, _Tp, _Compare, _Alloc >::upper_bound (
            const _Kt & __x) const -> decltype(const_iterator(_M_t._M_upper_bound_tr(__x)))
[inline]
```

Finds the end of a subsequence matching given key.

Parameters

_↩ _x	Key of (key, value) pair to be located.

Returns

Read-only (constant) iterator pointing to first iterator greater than key, or end().

upper_bound() [3/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::upper_bound (
            const key_type & __x)  [inline]
```

Finds the end of a subsequence matching given key.

Parameters

_↩ _x	Key of (key, value) pair to be located.

Returns

Iterator pointing to the first element greater than key, or end().

upper_bound() [4/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::upper_bound (
            const key_type & __x) const  [inline]
```

Finds the end of a subsequence matching given key.

Parameters

_↩ _x	Key of (key, value) pair to be located.

Returns

Read-only (constant) iterator pointing to first iterator greater than key, or end().

value_comp()

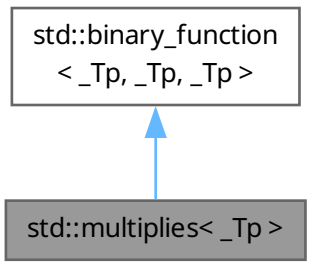
```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
value_compare std::multimap< _Key, _Tp, _Compare, _Alloc >::value_comp () const  [inline]
```

Returns a value comparison object, built from the key comparison object out of which the multimap was constructed.
The documentation for this class was generated from the following files:

- stl_map.h
- stl_multimap.h

5.684 std::multiplies< _Tp > Struct Template Reference

`#include <stl_function.h>`
Inheritance diagram for std::multiplies< _Tp >:



Public Types

- typedef _Tp first_argument_type
- typedef _Tp result_type
- typedef _Tp second_argument_type

Public Member Functions

- constexpr _Tp **operator()** (const _Tp &__x, const _Tp &__y) const

5.684.1 Detailed Description

template<**typename _Tp**>
struct std::multiplies< **_Tp** >

One of the math functors.

5.684.2 Member Typedef Documentation

first_argument_type

`typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::first_argument_type [inherited]`
`first_argument_type` is the type of the first argument

result_type

`typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::result_type [inherited]`
`result_type` is the return type

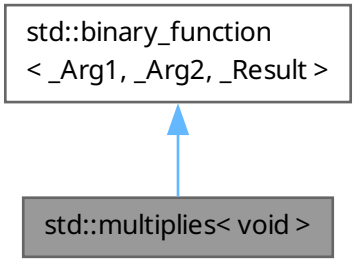
second_argument_type

`typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::second_argument_type [inherited]`
`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

5.685 std::multiplies< void > Struct Reference

`#include <stl_function.h>`
Inheritance diagram for std::multiplies< void >:



Public Types

- typedef void first_argument_type
- typedef __is_transparent **is_transparent**

- typedef void result_type
- typedef void second_argument_type

Public Member Functions

- template<typename _Tp, typename _Up>
  constexpr auto **operator()** (_Tp &&__t, _Up &&__u) const noexcept(noexcept(std::forward< _Tp >(__t) *std::forward< _Up >(__u))) -> decltype(std::forward< _Tp >(__t) *std::forward< _Up >(__u))
- constexpr void **operator()** (const void &__x, const void &__y) const

5.685.1 Detailed Description

One of the math functors.

5.685.2 Member Typedef Documentation

first_argument_type

`typedef void std::binary_function< void, void, void >::first_argument_type`
`first_argument_type` is the type of the first argument

result_type

`typedef void std::binary_function< void, void, void >::result_type`
`result_type` is the return type

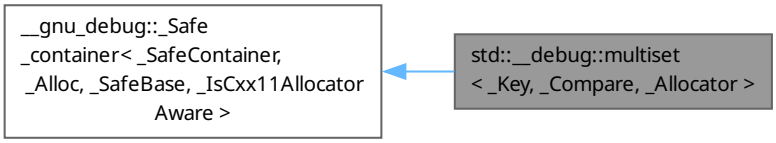
second_argument_type

`typedef void std::binary_function< void, void, void >::second_argument_type`
`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

5.686 std::__debug::multiset< _Key, _Compare, _Allocator > Class Template Reference

`#include <multiset.h>`
Inheritance diagram for std::__debug::multiset< _Key, _Compare, _Allocator >:



Public Types

- typedef _Allocator **allocator_type**
- typedef __gnu_debug::_Safe_iterator< _Base_const_iterator, multiset > **const_iterator**
- typedef _Base::const_pointer **const_pointer**

- typedef _Base::const_reference **const_reference**
- typedef std::reverse_iterator< const_iterator > **const_reverse_iterator**
- typedef _Base::difference_type **difference_type**
- typedef __gnu_debug::_Safe_iterator< _Base_iterator, multiset > **iterator**
- typedef _Compare **key_compare**
- typedef _Key **key_type**
- using **node_type**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef std::reverse_iterator< iterator > **reverse_iterator**
- typedef _Base::size_type **size_type**
- typedef _Compare **value_compare**
- typedef _Key **value_type**

Public Member Functions

- **multiset** (_Base_ref __x)
- template<typename _InputIterator>
  multiset (_InputIterator __first, _InputIterator __last, const _Compare &__comp=_Compare(), const _Allocator &__a=_Allocator())
- template<typename _InputIterator>
  multiset (_InputIterator __first, _InputIterator __last, const allocator_type &__a)
- **multiset** (const _Compare &__comp, const _Allocator &__a=_Allocator())
- **multiset** (const allocator_type &__a)
- **multiset** (const multiset &)=default
- **multiset** (const multiset &__m, const __type_identity_t< allocator_type > &__a)
- **multiset** (initializer_list< value_type > __l, const _Compare &__comp=_Compare(), const allocator_type &__↩ a=allocator_type())
- **multiset** (initializer_list< value_type > __l, const allocator_type &__a)
- **multiset** (multiset &&)=default
- **multiset** (multiset &&__m, const __type_identity_t< allocator_type > &__a) noexcept(noexcept(_Base(std::move(↩ __m), __a)))
- **__attribute** ((__abi_tag__("cxx11"))) iterator erase(const_iterator __first
- **__attribute** ((__abi_tag__("cxx11"))) iterator erase(const_iterator __position)
- const _Base & **_M_base** () const noexcept
- _Base & **_M_base** () noexcept
- const_iterator **begin** () const noexcept
- iterator **begin** () noexcept
- const_iterator **cbegin** () const noexcept
- const_iterator **cend** () const noexcept
- void **clear** () noexcept
- const_reverse_iterator **crbegin** () const noexcept
- const_reverse_iterator **crend** () const noexcept
- template<typename... _Args>
  iterator **emplace** (_Args &&... __args)
- template<typename... _Args>
  iterator **emplace_hint** (const_iterator __pos, _Args &&... __args)
- const_iterator **end** () const noexcept
- iterator **end** () noexcept
- template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>
  std::pair< iterator, iterator > **equal_range** (const _Kt &__x)

- template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>
  std::pair< const_iterator, const_iterator > **equal_range** (const _Kt &__x) const
- std::pair< iterator, iterator > **equal_range** (const key_type &__x)
- std::pair< const_iterator, const_iterator > **equal_range** (const key_type &__x) const
- _Base_iterator **erase** (_Base_const_iterator __position)
- size_type **erase** (const key_type &__x)
- node_type **extract** (const key_type &__key)
- node_type **extract** (const_iterator __position)
- template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>
  iterator **find** (const _Kt &__x)
- template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>
  const_iterator **find** (const _Kt &__x) const
- iterator **find** (const key_type &__x)
- const_iterator **find** (const key_type &__x) const
- **for** (_Base_const_iterator __victim=__first.base();__victim !=__last.base();++__victim)
- template<typename _InputIterator>
  void **insert** (_InputIterator __first, _InputIterator __last)
- iterator **insert** (const value_type &__x)
- iterator **insert** (const_iterator __hint, node_type &&__nh)
- iterator **insert** (const_iterator __position, const value_type &__x)
- iterator **insert** (const_iterator __position, value_type &&__x)
- void **insert** (initializer_list< value_type > __l)
- iterator **insert** (node_type &&__nh)
- iterator **insert** (value_type &&__x)
- template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>
  iterator **lower_bound** (const _Kt &__x)
- template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>
  const_iterator **lower_bound** (const _Kt &__x) const
- iterator **lower_bound** (const key_type &__x)
- const_iterator **lower_bound** (const key_type &__x) const
- multiset & **operator=** (const multiset &)=default
- multiset & **operator=** (initializer_list< value_type > __l)
- multiset & **operator=** (multiset &&)=default
- const_reverse_iterator **rbegin** () const noexcept
- reverse_iterator **rbegin** () noexcept
- const_reverse_iterator **rend** () const noexcept
- reverse_iterator **rend** () noexcept
- void **swap** (multiset &__x) noexcept(/*conditional */)
- template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>
  iterator **upper_bound** (const _Kt &__x)
- template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>
  const_iterator **upper_bound** (const _Kt &__x) const
- iterator **upper_bound** (const key_type &__x)
- const_iterator **upper_bound** (const key_type &__x) const
- **while** (false)

Public Attributes

- const_iterator **__last**
- **do**
- **return**

Protected Member Functions

- constexpr void **_M_swap** (const _Safe_container &__x) const noexcept

Friends

- template<typename _ItT, typename _SeqT, typename _CatT>
  class **::__gnu_debug::_Safe_iterator**

5.686.1 Detailed Description

template<typename _Key, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<_Key>>
class std::__debug::multiset< _Key, _Compare, _Allocator >

Class std::multiset with safety/checking/debug instrumentation.
The documentation for this class was generated from the following file:

- multiset.h

5.687 std::multiset< _Key, _Compare, _Alloc > Class Template Reference

`#include <set>`

Public Types

- typedef _Alloc **allocator_type**
- typedef _Rep_type::const_iterator **const_iterator**
- typedef _Alloc_traits::const_pointer **const_pointer**
- typedef _Alloc_traits::const_reference **const_reference**
- typedef _Rep_type::const_reverse_iterator **const_reverse_iterator**
- typedef _Rep_type::difference_type **difference_type**
- typedef _Rep_type::const_iterator **iterator**
- typedef _Compare **key_compare**
- typedef _Key **key_type**
- typedef _Alloc_traits::pointer **pointer**
- typedef _Alloc_traits::reference **reference**
- typedef _Rep_type::const_reverse_iterator **reverse_iterator**
- typedef _Rep_type::size_type **size_type**
- typedef _Compare **value_compare**
- typedef _Key **value_type**

Public Member Functions

- multiset ()=default
- template<typename _InputIterator>
  multiset (_InputIterator __first, _InputIterator __last)
- template<typename _InputIterator>
  multiset (_InputIterator __first, _InputIterator __last, const _Compare &__comp, const allocator_type &__a=allocator_type())
- template<typename _InputIterator>
  multiset (_InputIterator __first, _InputIterator __last, const allocator_type &__a)
- multiset (const _Compare &__comp, const allocator_type &__a=allocator_type())
- multiset (const allocator_type &__a)

- multiset (const multiset &)=default
- multiset (const multiset &__m, const __type_identity_t< allocator_type > &__a)
- multiset (initializer_list< value_type > __l, const _Compare &__comp=_Compare(), const allocator_type &__a=allocator_type())
- multiset (initializer_list< value_type > __l, const allocator_type &__a)
- multiset (multiset &&)=default
- multiset (multiset &&__m, const __type_identity_t< allocator_type > &__a) noexcept(is_nothrow_copy_constructible< _Compare >::value &&_Alloc_traits::_S_always_equal())
- ~multiset ()=default
- iterator begin () const noexcept
- iterator cbegin () const noexcept
- iterator cend () const noexcept
- void clear () noexcept
- reverse_iterator crbegin () const noexcept
- reverse_iterator crend () const noexcept
- template<typename... _Args>
  iterator emplace (_Args &&... __args)
- template<typename... _Args>
  iterator emplace_hint (const_iterator __pos, _Args &&... __args)
- bool empty () const noexcept
- iterator end () const noexcept
- size_type erase (const key_type &__x)
- _GLIBCXX_ABI_TAG_CXX11 iterator erase (const_iterator __first, const_iterator __last)
- _GLIBCXX_ABI_TAG_CXX11 iterator erase (const_iterator __position)
- allocator_type get_allocator () const noexcept
- template<typename _InputIterator>
  void insert (_InputIterator __first, _InputIterator __last)
- iterator insert (const value_type &__x)
- iterator insert (const_iterator __position, const value_type &__x)
- iterator **insert** (const_iterator __position, value_type &&__x)
- void insert (initializer_list< value_type > __l)
- iterator **insert** (value_type &&__x)
- key_compare key_comp () const
- size_type max_size () const noexcept
- multiset & operator= (const multiset &)=default
- multiset & operator= (initializer_list< value_type > __l)
- multiset & operator= (multiset &&)=default
- reverse_iterator rbegin () const noexcept
- reverse_iterator rend () const noexcept
- size_type size () const noexcept
- void swap (multiset &__x) noexcept(/*conditional */)
- value_compare value_comp () const

- size_type count (const key_type &__x) const
- template<typename _Kt>
  auto count (const _Kt &__x) const -> decltype(_M_t._M_count_tr(__x))

- bool contains (const key_type &__x) const

- template<typename _Kt>
  auto contains (const _Kt &__x) const -> decltype(_M_t._M_find_tr(__x), void(), true)

- iterator find (const key_type &__x)
- const_iterator find (const key_type &__x) const
- template<typename _Kt>
  auto find (const _Kt &__x) -> decltype(iterator{_M_t._M_find_tr(__x)})
- template<typename _Kt>
  auto find (const _Kt &__x) const -> decltype(const_iterator{_M_t._M_find_tr(__x)})

- iterator lower_bound (const key_type &__x)
- const_iterator lower_bound (const key_type &__x) const
- template<typename _Kt>
  auto lower_bound (const _Kt &__x) -> decltype(iterator(_M_t._M_lower_bound_tr(__x)))
- template<typename _Kt>
  auto lower_bound (const _Kt &__x) const -> decltype(iterator(_M_t._M_lower_bound_tr(__x)))

- iterator upper_bound (const key_type &__x)
- const_iterator upper_bound (const key_type &__x) const
- template<typename _Kt>
  auto upper_bound (const _Kt &__x) -> decltype(iterator(_M_t._M_upper_bound_tr(__x)))
- template<typename _Kt>
  auto upper_bound (const _Kt &__x) const -> decltype(iterator(_M_t._M_upper_bound_tr(__x)))

- std::pair< iterator, iterator > equal_range (const key_type &__x)
- std::pair< const_iterator, const_iterator > equal_range (const key_type &__x) const
- template<typename _Kt>
  auto equal_range (const _Kt &__x) -> decltype(pair< iterator, iterator >(_M_t._M_equal_range_tr(__x)))
- template<typename _Kt>
  auto equal_range (const _Kt &__x) const -> decltype(pair< iterator, iterator >(_M_t._M_equal_range_tr(__x)))

Friends

- template<typename _K1, typename _C1, typename _A1>
  bool **operator**< (const multiset< _K1, _C1, _A1 > &, const multiset< _K1, _C1, _A1 > &)
- template<typename _K1, typename _C1, typename _A1>
  bool **operator==** (const multiset< _K1, _C1, _A1 > &, const multiset< _K1, _C1, _A1 > &)

5.687.1 Detailed Description

template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
class std::multiset< _Key, _Compare, _Alloc >

A standard container made up of elements, which can be retrieved in logarithmic time.

Since

C++98

Template Parameters

_Key	Type of key objects.
_Compare	Comparison function object type, defaults to less<_Key>.
_Alloc	Allocator type, defaults to allocator<_Key>.

Meets the requirements of a container, a reversible container, and an associative container (using equivalent keys). For a multiset<Key> the key_type and value_type are Key.
Multisets support bidirectional iterators.
The private tree data is declared exactly the same way for set and multiset; the distinction is made entirely in how the tree functions are called (*_unique versus *_equal, same as the standard).

5.687.2 Constructor & Destructor Documentation

multiset() [1/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset ()  [default]
```

Default constructor creates no elements.

multiset() [2/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset (
            const _Compare & __comp,
            const allocator_type & __a = allocator_type())  [inline], [explicit]
```

Creates a multiset with no elements.

Parameters

__comp	Comparator to use.
__a	An allocator object.

multiset() [3/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
template<typename _InputIterator>
std::multiset< _Key, _Compare, _Alloc >::multiset (
            _InputIterator __first,
            _InputIterator __last)  [inline]
```

Builds a multiset from a range.

Parameters

__first	An input iterator.
__last	An input iterator.

Create a multiset consisting of copies of the elements from [first,last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(__first,__last)).

multiset() [4/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
template<typename _InputIterator>
std::multiset< _Key, _Compare, _Alloc >::multiset (
            _InputIterator __first,
            _InputIterator __last,
            const _Compare & __comp,
            const allocator_type & __a = allocator_type())  [inline]
```

Builds a multiset from a range.

Parameters

__first	An input iterator.
__last	An input iterator.
__comp	A comparison functor.
__a	An allocator object.

Create a multiset consisting of copies of the elements from [__first,__last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(__first,__last)).

multiset() [5/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset (
            const multiset< _Key, _Compare, _Alloc > & )  [default]
```

Multiset copy constructor.
Whether the allocator is copied depends on the allocator traits.

multiset() [6/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset (
            multiset< _Key, _Compare, _Alloc > && )  [default]
```

Multiset move constructor.
The newly-created multiset contains the exact contents of the moved instance. The moved instance is a valid, but unspecified multiset.

multiset() [7/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset (
            initializer_list< value_type > __l,
            const _Compare & __comp = _Compare(),
            const allocator_type & __a = allocator_type())  [inline]
```

Builds a multiset from an initializer_list.

Parameters

__l	An initializer_list.

Parameters

__comp	A comparison functor.
__a	An allocator object.

Create a multiset consisting of copies of the elements from the list. This is linear in N if the list is already sorted, and NlogN otherwise (where N is *__l.size()*).

multiset() [8/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset (
            const allocator_type & __a)  [inline], [explicit]
```

Allocator-extended default constructor.

multiset() [9/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset (
            const multiset< _Key, _Compare, _Alloc > & __m,
            const __type_identity_t< allocator_type > & __a)  [inline]
```

Allocator-extended copy constructor.

multiset() [10/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset (
            multiset< _Key, _Compare, _Alloc > && __m,
            const __type_identity_t< allocator_type > & __a)  [inline], [noexcept]
```

Allocator-extended move constructor.

multiset() [11/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset (
            initializer_list< value_type > __l,
            const allocator_type & __a)  [inline]
```

Allocator-extended initialier-list constructor.

multiset() [12/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
template<typename _InputIterator>
std::multiset< _Key, _Compare, _Alloc >::multiset (
            _InputIterator __first,
            _InputIterator __last,
            const allocator_type & __a)  [inline]
```

Allocator-extended range constructor.

~multiset()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::multiset< _Key, _Compare, _Alloc >::~multiset ()  [default]
```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

5.687.3 Member Function Documentation

begin()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::begin () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the multiset. Iteration is done in ascending order according to the keys.

cbegin()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::cbegin () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the multiset. Iteration is done in ascending order according to the keys.

cend()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::cend () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the multiset. Iteration is done in ascending order according to the keys.

clear()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
void std::multiset< _Key, _Compare, _Alloc >::clear ()  [inline], [noexcept]
```

Erases all elements in a multiset. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

contains() [1/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
template<typename _Kt>
auto std::multiset< _Key, _Compare, _Alloc >::contains (
            const _Kt & __x) const -> decltype(_M_t._M_find_tr(__x), void(), true)   [inline]
```

Finds whether an element with the given key exists.

Parameters

_↩ _x	Key of elements to be located.

Returns

True if there is any element with the specified key.

contains() [2/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
bool std::multiset< _Key, _Compare, _Alloc >::contains (
            const key_type & __x) const  [inline]
```

Finds whether an element with the given key exists.

Parameters

_↩ _x	Key of elements to be located.

Returns

True if there is any element with the specified key.

count() [1/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
template<typename _Kt>
auto std::multiset< _Key, _Compare, _Alloc >::count (
            const _Kt & __x) const -> decltype(_M_t._M_count_tr(__x))   [inline]
```

Finds the number of elements with given key.

Parameters

_↩ _x	Key of elements to be located.

Returns

Number of elements with specified key.

count() [2/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
size_type std::multiset< _Key, _Compare, _Alloc >::count (
            const key_type & __x) const  [inline]
```

Finds the number of elements with given key.

Parameters

_↩ _x	Key of elements to be located.

Returns

Number of elements with specified key.

crbegin()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
reverse_iterator std::multiset< _Key, _Compare, _Alloc >::crbegin () const  [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

crend()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
reverse_iterator std::multiset< _Key, _Compare, _Alloc >::crend () const  [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

emplace()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
template<typename... _Args>
iterator std::multiset< _Key, _Compare, _Alloc >::emplace (
            _Args &&... __args)  [inline]
```

Builds and inserts an element into the multiset.

Parameters

__args	Arguments used to generate the element instance to be inserted.

Returns

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a std::set the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.
Insertion requires logarithmic time.

emplace_hint()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
template<typename... _Args>
iterator std::multiset< _Key, _Compare, _Alloc >::emplace_hint (
            const_iterator __pos,
            _Args &&... __args)  [inline]
```

Builds and inserts an element into the multiset.

Parameters

__pos	An iterator that serves as a hint as to where the element should be inserted.
__args	Arguments used to generate the element instance to be inserted.

Returns

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a std::set the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.
Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
See https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints for more on *hinting*.
Insertion requires logarithmic time (if the hint is not taken).

empty()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
bool std::multiset< _Key, _Compare, _Alloc >::empty () const  [inline], [nodiscard], [noexcept]
```

Returns true if the set is empty.

end()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::end () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the multiset. Iteration is done in ascending order according to the keys.

equal_range() [1/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _Kt>
auto std::multiset< _Key, _Compare, _Alloc >::equal_range (
            const _Kt & __x) -> decltype(pair<iterator, iterator>(_M_t._M_equal_range_tr(__x)))
[inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
               c.upper_bound(val))
```

(but is faster than making the calls separately).
This function probably only makes sense for multisets.

equal_range() [2/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
```

```
template<typename _Kt>
auto std::multiset< _Key, _Compare, _Alloc >::equal_range (
            const _Kt & __x) const -> decltype(pair<iterator, iterator>(_M_t._M_equal_range_↩
tr(__x)))   [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
               c.upper_bound(val))
```

(but is faster than making the calls separately).
This function probably only makes sense for multisets.

equal_range() [3/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::pair< iterator, iterator > std::multiset< _Key, _Compare, _Alloc >::equal_range (
            const key_type & __x)  [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
               c.upper_bound(val))
```

(but is faster than making the calls separately).
This function probably only makes sense for multisets.

equal_range() [4/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::pair< const_iterator, const_iterator > std::multiset< _Key, _Compare, _Alloc >::equal_range
(
            const key_type & __x) const  [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
               c.upper_bound(val))
```

(but is faster than making the calls separately).
This function probably only makes sense for multisets.

erase() [1/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
size_type std::multiset< _Key, _Compare, _Alloc >::erase (
            const key_type & __x)  [inline]
```

Erases elements according to the provided key.

Parameters

_↩ _x	Key of element to be erased.

Returns

The number of elements erased.

This function erases all elements located by the given key from a multiset. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [2/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
_GLIBCXX_ABI_TAG_CXX11 iterator std::multiset< _Key, _Compare, _Alloc >::erase (
            const_iterator __first,
            const_iterator __last)  [inline]
```

Erases a [first,last) range of elements from a multiset.

Parameters

__first	Iterator pointing to the start of the range to be erased.
__last	Iterator pointing to the end of the range to be erased.

Returns

The iterator *last*.

This function erases a sequence of elements from a multiset. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [3/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
_GLIBCXX_ABI_TAG_CXX11 iterator std::multiset< _Key, _Compare, _Alloc >::erase (
            const_iterator __position)  [inline]
```

Erases an element from a multiset.

Parameters

__position	An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from a multiset. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

find() [1/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
template<typename _Kt>
auto std::multiset< _Key, _Compare, _Alloc >::find (
            const _Kt & __x) -> decltype(iterator{_M_t._M_find_tr(__x)})   [inline]
```

Tries to locate an element in a set.

Parameters

_↩ _*x*	Element to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

find() [2/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
template<typename _Kt>
auto std::multiset< _Key, _Compare, _Alloc >::find (
            const _Kt & __x) const -> decltype(const_iterator{_M_t._M_find_tr(__x)})   [inline]
```

Tries to locate an element in a set.

Parameters

_↩ _*x*	Element to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

find() [3/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::find (
            const key_type & __x)  [inline]
```

Tries to locate an element in a set.

Parameters

_↩ _x	Element to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

find() [4/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
const_iterator std::multiset< _Key, _Compare, _Alloc >::find (
            const key_type & __x) const  [inline]
```

Tries to locate an element in a set.

Parameters

_↩ _x	Element to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

get_allocator()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
allocator_type std::multiset< _Key, _Compare, _Alloc >::get_allocator () const  [inline], [noexcept]
```

Returns the memory allocation object.

insert() [1/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
template<typename _InputIterator>
void std::multiset< _Key, _Compare, _Alloc >::insert (
            _InputIterator __first,
            _InputIterator __last)  [inline]
```

A template function that tries to insert a range of elements.

Parameters

__first	Iterator pointing to the start of the range to be inserted.
__last	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

insert() [2/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::insert (
            const value_type & __x)  [inline]
```

Inserts an element into the multiset.

Parameters

_↩ _x	Element to be inserted.

Returns

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a std::set the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.
Insertion requires logarithmic time.
Referenced by std::multiset< _Key, _Cmp, polymorphic_allocator< _Key > >::insert().

insert() [3/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::insert (
            const_iterator __position,
            const value_type & __x)  [inline]
```

Inserts an element into the multiset.

Parameters

__position	An iterator that serves as a hint as to where the element should be inserted.
__x	Element to be inserted.

Returns

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a std::set the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.
Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
See https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints for more on *hinting*.
Insertion requires logarithmic time (if the hint is not taken).

insert() [4/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
void std::multiset< _Key, _Compare, _Alloc >::insert (
            initializer_list< value_type > __l)  [inline]
```

Attempts to insert a list of elements into the multiset.

Parameters

__l	A std::initializer_list<value_type> of elements to be inserted.

Complexity similar to that of the range constructor.

key_comp()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
key_compare std::multiset< _Key, _Compare, _Alloc >::key_comp () const  [inline]
```

Returns the comparison object.

lower_bound() [1/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _Kt>
auto std::multiset< _Key, _Compare, _Alloc >::lower_bound (
            const _Kt & __x) -> decltype(iterator(_M_t._M_lower_bound_tr(__x)))   [inline]
```

Finds the beginning of a subsequence matching given key.

Parameters

__x	Key to be located.

Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

lower_bound() [2/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
template<typename _Kt>
auto std::multiset< _Key, _Compare, _Alloc >::lower_bound (
            const _Kt & __x) const -> decltype(iterator(_M_t._M_lower_bound_tr(__x)))   [inline]
```

Finds the beginning of a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

lower_bound() [3/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::lower_bound (
            const key_type & __x)  [inline]
```

Finds the beginning of a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

lower_bound() [4/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
const_iterator std::multiset< _Key, _Compare, _Alloc >::lower_bound (
            const key_type & __x) const  [inline]
```

Finds the beginning of a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

max_size()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
size_type std::multiset< _Key, _Compare, _Alloc >::max_size () const  [inline], [noexcept]
```

Returns the maximum size of the set.

operator=() [1/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
multiset & std::multiset< _Key, _Compare, _Alloc >::operator= (
            const multiset< _Key, _Compare, _Alloc > & )  [default]
```

Multiset assignment operator.
Whether the allocator is copied depends on the allocator traits.

operator=() [2/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
multiset & std::multiset< _Key, _Compare, _Alloc >::operator= (
            initializer_list< value_type > __l)  [inline]
```

Multiset list assignment operator.

Parameters

↩ _↩ ↩ _↩ *l*	An initializer_list.

This function fills a multiset with copies of the elements in the initializer list *__l*.
Note that the assignment completely changes the multiset and that the resulting multiset's size is the same as the number of elements assigned.

operator=() [3/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
multiset & std::multiset< _Key, _Compare, _Alloc >::operator= (
            multiset< _Key, _Compare, _Alloc > && )  [default]
```

Move assignment operator.

rbegin()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
reverse_iterator std::multiset< _Key, _Compare, _Alloc >::rbegin () const  [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

rend()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
reverse_iterator std::multiset< _Key, _Compare, _Alloc >::rend () const  [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

size()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
size_type std::multiset< _Key, _Compare, _Alloc >::size () const  [inline], [noexcept]
```

Returns the size of the set.

swap()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
void std::multiset< _Key, _Compare, _Alloc >::swap (
            multiset< _Key, _Compare, _Alloc > & __x)  [inline], [noexcept]
```

Swaps data with another multiset.

Parameters

_↩ _x	A multiset of the same element and allocator types.

This exchanges the elements between two multisets in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global std::swap() function is specialized such that std::swap(s1,s2) will feed to this function.
Whether the allocators are swapped depends on the allocator traits.
Referenced by std::swap().

upper_bound() [1/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
template<typename _Kt>
auto std::multiset< _Key, _Compare, _Alloc >::upper_bound (
            const _Kt & __x) -> decltype(iterator(_M_t._M_upper_bound_tr(__x)))  [inline]
```

Finds the end of a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Iterator pointing to the first element greater than key, or end().

upper_bound() [2/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
template<typename _Kt>
auto std::multiset< _Key, _Compare, _Alloc >::upper_bound (
            const _Kt & __x) const -> decltype(iterator(_M_t._M_upper_bound_tr(__x)))   [inline]
```

Finds the end of a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Iterator pointing to the first element greater than key, or end().

upper_bound() [3/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::upper_bound (
            const key_type & __x)  [inline]
```

Finds the end of a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Iterator pointing to the first element greater than key, or end().

upper_bound() [4/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
const_iterator std::multiset< _Key, _Compare, _Alloc >::upper_bound (
            const key_type & __x) const  [inline]
```

Finds the end of a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Iterator pointing to the first element greater than key, or end().

value_comp()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
value_compare std::multiset< _Key, _Compare, _Alloc >::value_comp () const  [inline]
```

Returns the comparison object.

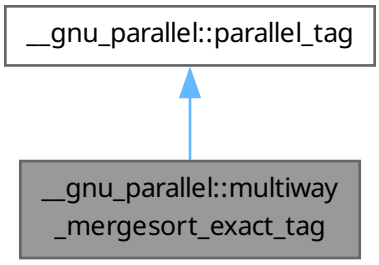
The documentation for this class was generated from the following file:

- stl_multiset.h

5.688 __gnu_parallel::multiway_mergesort_exact_tag Struct Reference

```
#include <tags.h>
```

Inheritance diagram for __gnu_parallel::multiway_mergesort_exact_tag:



Public Member Functions

- **multiway_mergesort_exact_tag** (_ThreadIndex __num_threads)
- _ThreadIndex __get_num_threads ()
- void set_num_threads (_ThreadIndex __num_threads)

5.688.1 Detailed Description

Forces parallel sorting using multiway mergesort with exact splitting at compile time.

5.688.2 Member Function Documentation

__get_num_threads()

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ()  [inline], [inherited]
```

Find out desired number of threads.

Returns

Desired number of threads.

Referenced by __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), and __gnu_parallel::__parallel_sort(

set_num_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
            _ThreadIndex __num_threads)  [inline], [inherited]
```

Set the desired number of threads.

Parameters

__num_threads	Desired number of threads.

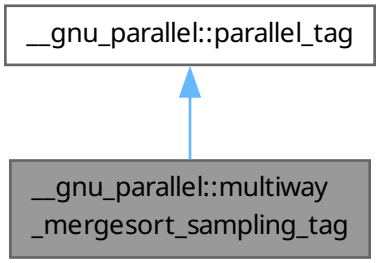
The documentation for this struct was generated from the following file:

- tags.h

5.689 __gnu_parallel::multiway_mergesort_sampling_tag Struct Reference

```
#include <tags.h>
```

Inheritance diagram for __gnu_parallel::multiway_mergesort_sampling_tag:



Public Member Functions

- **multiway_mergesort_sampling_tag** (_ThreadIndex __num_threads)
- _ThreadIndex __get_num_threads ()
- void set_num_threads (_ThreadIndex __num_threads)

5.689.1 Detailed Description

Forces parallel sorting using multiway mergesort with splitting by sampling at compile time.

5.689.2 Member Function Documentation

__get_num_threads()

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ()  [inline], [inherited]
```

Find out desired number of threads.

Returns

Desired number of threads.

Referenced by __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), and __gnu_parallel::__parallel_sort(

set_num_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
            _ThreadIndex __num_threads)  [inline], [inherited]
```

Set the desired number of threads.

Parameters

__num_threads	Desired number of threads.

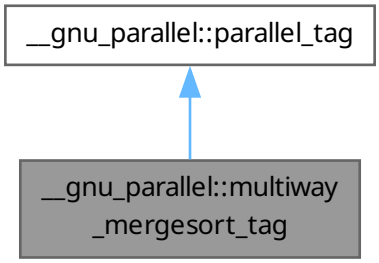
The documentation for this struct was generated from the following file:

- tags.h

5.690 __gnu_parallel::multiway_mergesort_tag Struct Reference

```
#include <tags.h>
```

Inheritance diagram for __gnu_parallel::multiway_mergesort_tag:



Public Member Functions

- **multiway_mergesort_tag** (_ThreadIndex __num_threads)
- _ThreadIndex __get_num_threads ()
- void set_num_threads (_ThreadIndex __num_threads)

5.690.1 Detailed Description

Forces parallel sorting using multiway mergesort at compile time.

5.690.2 Member Function Documentation

__get_num_threads()

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ()  [inline], [inherited]
```

Find out desired number of threads.

Returns

Desired number of threads.

Referenced by __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), and __gnu_parallel::__parallel_sort(

set_num_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
            _ThreadIndex __num_threads)  [inline], [inherited]
```

Set the desired number of threads.

Parameters

__num_threads	Desired number of threads.

The documentation for this struct was generated from the following file:

- tags.h

5.691 std::mutex Class Reference

```
#include <mutex>
```

Public Types

- typedef __native_type * **native_handle_type**

Public Member Functions

- **mutex** (const mutex &)=delete
- void **lock** ()
- native_handle_type **native_handle** () noexcept
- mutex & **operator=** (const mutex &)=delete
- bool **try_lock** () noexcept
- void **unlock** ()

5.691.1 Detailed Description

The standard mutex type.
A simple, non-recursive, non-timed mutex.
Do not call `lock()` and `unlock()` directly, use a scoped lock type such as `std::unique_lock`, `std::lock_guard`, or (since C++17) `std::scoped_lock`.

Since

C++11

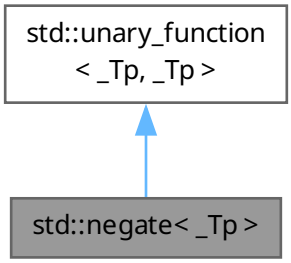
The documentation for this class was generated from the following file:

- std_mutex.h

5.692 std::negate< _Tp > Struct Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::negate< _Tp >:



Public Types

- typedef _Tp argument_type
- typedef _Tp result_type

Public Member Functions

- constexpr _Tp **operator()** (const _Tp &__x) const

5.692.1 Detailed Description

template<**typename _Tp**>
struct std::negate< **_Tp** >

One of the math functors.

5.692.2 Member Typedef Documentation

argument_type

```
typedef _Tp std::unary_function< _Tp, _Tp >::argument_type  [inherited]
```

`argument_type` is the type of the argument

result_type

```
typedef _Tp std::unary_function< _Tp, _Tp >::result_type  [inherited]
```

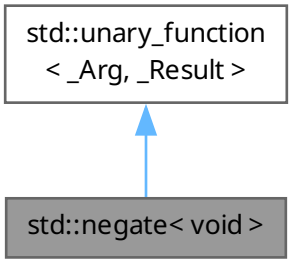
`result_type` is the return type
The documentation for this struct was generated from the following file:

- stl_function.h

5.693 std::negate< void > Struct Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::negate< void >:



Public Types

- typedef void argument_type
- typedef __is_transparent **is_transparent**
- typedef void result_type

Public Member Functions

- template<typename _Tp>
  constexpr auto **operator()** (_Tp &&__t) const noexcept(noexcept(-std::forward< _Tp >(__t))) -> decltype(-std::forward< _Tp >(__t))
- constexpr void **operator()** (const void &__x) const

5.693.1 Detailed Description

One of the math functors.

5.693.2 Member Typedef Documentation

argument_type

```
typedef void std::unary_function< void, void >::argument_type
```

`argument_type` is the type of the argument

result_type

```
typedef void std::unary_function< void, void >::result_type
```

`result_type` is the return type

The documentation for this struct was generated from the following file:

- stl_function.h

5.694 std::negative_binomial_distribution< _IntType > Class Template Reference

```
#include <random>
```

Classes

- struct param_type

Public Types

- typedef _IntType result_type

Public Member Functions

- **negative_binomial_distribution** (_IntType __k, double __p=0.5)
- **negative_binomial_distribution** (const param_type &__p)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- _IntType k () const
- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- double p () const
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- template<typename _IntType1, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::negative_binomial_distribution< _IntType1 > &__x)
- bool operator== (const negative_binomial_distribution &__d1, const negative_binomial_distribution &__d2)
- template<typename _IntType1, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::negative_binomial_distribution< _IntType1 > &__x)

5.694.1 Detailed Description

template<typename _IntType = int>
class std::negative_binomial_distribution< _IntType >

A negative_binomial_distribution random number distribution.
The formula for the negative binomial probability mass function is $p(i) = \binom{n}{i} p^i (1-p)^{t-i}$ where t and p are the parameters of the distribution.

Since

C++11

5.694.2 Member Typedef Documentation

result_type

```
template<typename _IntType = int>
typedef _IntType std::negative_binomial_distribution< _IntType >::result_type
```

The type of the range of the distribution.

5.694.3 Member Function Documentation

k()

```
template<typename _IntType = int>
_IntType std::negative_binomial_distribution< _IntType >::k () const  [inline]
```

Return the k parameter of the distribution.

max()

```
template<typename _IntType = int>
result_type std::negative_binomial_distribution< _IntType >::max () const  [inline]
```

Returns the least upper bound value of the distribution.

min()

```
template<typename _IntType = int>
result_type std::negative_binomial_distribution< _IntType >::min () const  [inline]
```

Returns the greatest lower bound value of the distribution.

operator()()

```
template<typename _IntType>
template<typename _UniformRandomNumberGenerator>
negative_binomial_distribution< _IntType >::result_type std::negative_binomial_distribution< _↩
IntType >::operator() (
            _UniformRandomNumberGenerator & __urng)
```

Generating functions.

p()

```
template<typename _IntType = int>
double std::negative_binomial_distribution< _IntType >::p () const  [inline]
```

Return the p parameter of the distribution.

param() [1/2]

```
template<typename _IntType = int>
param_type std::negative_binomial_distribution< _IntType >::param () const  [inline]
```

Returns the parameter set of the distribution.

param() [2/2]

```
template<typename _IntType = int>
void std::negative_binomial_distribution< _IntType >::param (
            const param_type & __param)  [inline]
```

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

reset()

```
template<typename _IntType = int>
void std::negative_binomial_distribution< _IntType >::reset ()  [inline]
```

Resets the distribution state.

5.694.4 Friends And Related Symbol Documentation

operator<<

```
template<typename _IntType = int>
template<typename _IntType1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::negative_binomial_distribution< _IntType1 > & __x)  [friend]
```

Inserts a negative_binomial_distribution random number distribution __x into the output stream __os.

Parameters

__os	An output stream.
__x	A negative_binomial_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

operator==

```
template<typename _IntType = int>
bool operator== (
            const negative_binomial_distribution< _IntType > & __d1,
            const negative_binomial_distribution< _IntType > & __d2)  [friend]
```

Return true if two negative binomial distributions have the same parameters and the sequences that would be generated are equal.

operator>>

```
template<typename _IntType = int>
template<typename _IntType1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            std::negative_binomial_distribution< _IntType1 > & __x)  [friend]
```

Extracts a negative_binomial_distribution random number distribution __x from the input stream __is.

Parameters

_↩ _is	An input stream.

_↩ _x	A negative_binomial_distribution random number generator engine.

Returns

The input stream with __x extracted or in an error state.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

5.695 std::nested_exception Class Reference

`#include <exception>`

Public Member Functions

- nested_exception () noexcept
- **nested_exception** (const nested_exception &) noexcept=default
- exception_ptr nested_ptr () const noexcept
- nested_exception & **operator=** (const nested_exception &) noexcept=default
- void rethrow_nested () const

5.695.1 Detailed Description

Mixin class that stores the current exception.
This type can be used via `std::throw_with_nested` to store the current exception nested within another exception.

Since

C++11

See also

std::throw_with_nested

5.695.2 Constructor & Destructor Documentation

nested_exception()

`std::nested_exception::nested_exception () [inline], [noexcept]`
The default constructor stores the current exception (if any).
References std::current_exception().

5.695.3 Member Function Documentation

nested_ptr()

`exception_ptr std::nested_exception::nested_ptr () const [inline], [noexcept]`
Access the stored exception.

rethrow_nested()

```
void std::nested_exception::rethrow_nested () const  [inline]
```

Rethrow the stored exception, or terminate if none was stored.
References std::rethrow_exception(), and std::terminate().
The documentation for this class was generated from the following file:

- nested_exception.h

5.696 __gnu_cxx::limit_condition::never_adjustor Struct Reference

`#include <throw_allocator.h>`

5.696.1 Detailed Description

Never enter the condition.
The documentation for this struct was generated from the following file:

- throw_allocator.h

5.697 __gnu_cxx::random_condition::never_adjustor Struct Reference

`#include <throw_allocator.h>`

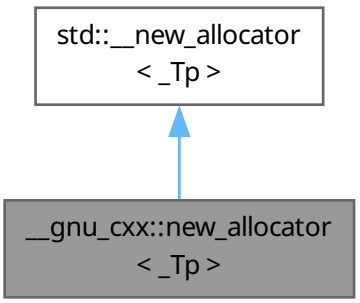
5.697.1 Detailed Description

Never enter the condition.
The documentation for this struct was generated from the following file:

- throw_allocator.h

5.698 __gnu_cxx::new_allocator< _Tp > Class Template Reference

`#include <ext/new_allocator.h>`
Inheritance diagram for __gnu_cxx::new_allocator< _Tp >:



Public Types

- typedef std::ptrdiff_t **difference_type**
- typedef std::true_type **propagate_on_container_move_assignment**
- typedef std::size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **new_allocator** (const new_allocator &) noexcept
- template<typename _Tp1>
  new_allocator (const new_allocator< _Tp1 > &) noexcept
- _Tp * **allocate** (size_type __n, const void *=static_cast< const void * >(0))
- void **deallocate** (_Tp *__p, size_type __n)

5.698.1 Detailed Description

template<typename _Tp>
class __gnu_cxx::new_allocator< _Tp >

An allocator that uses global `new`, as per C++03 [20.4.1].
This is precisely the allocator defined in the C++ Standard.

- all allocation calls `operator new`
- all deallocation calls `operator delete`

This is a non-standard extension that can be used to guarantee allocation from `new` even if the library has been configured to use a different implementation for `std::allocator`.

Template Parameters

_Tp	Type of allocated object.

The documentation for this class was generated from the following file:

- ext/new_allocator.h

5.699 __gnu_pbds::detail::no_throw_copies< Key, Mapped > Struct Template Reference

`#include <types_traits.hpp>`

Public Types

- typedef integral_constant< int, __simple > **indicator**

Static Public Attributes

- static const bool **__simple**

5.699.1 Detailed Description

template<typename Key, typename Mapped>
struct __gnu_pbds::detail::no_throw_copies< Key, Mapped >

Primary template.
The documentation for this struct was generated from the following file:

- types_traits.hpp

5.700 __gnu_pbds::detail::no_throw_copies< Key, null_type > Struct Template Reference

`#include <types_traits.hpp>`

Public Types

- typedef integral_constant< int, __simple > **indicator**
- typedef integral_constant< int, is_simple< Key >::value > **indicator**

Static Public Attributes

- static const bool **__simple**

5.700.1 Detailed Description

template<typename Key>
struct __gnu_pbds::detail::no_throw_copies< Key, null_type >

Specialization.
The documentation for this struct was generated from the following file:

- types_traits.hpp

5.701 std::normal_distribution< _RealType > Class Template Reference

`#include <random>`

Classes

- struct param_type

Public Types

- typedef _RealType result_type

Public Member Functions

- **normal_distribution** (const param_type &__p)
- normal_distribution (result_type __mean, result_type __stddev=result_type(1))
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- result_type max () const
- _RealType mean () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const

- void param (const param_type &__param)
- void reset ()
- _RealType stddev () const

Friends

- template<typename _RealType1, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::normal_distribution< _RealType1 > &__x)
- template<typename _RealType1>
  bool operator== (const std::normal_distribution< _RealType1 > &__d1, const std::normal_distribution< _Real↩ Type1 > &__d2)
- template<typename _RealType1, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::normal_distribution< _RealType1 > &__x)

5.701.1 Detailed Description

template<typename _RealType = double>
class std::normal_distribution< _RealType >

A normal continuous distribution for random numbers.
The formula for the normal probability density function is

$$p(x|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x-\mu^2}{2\sigma^2}}$$

Since

C++11

5.701.2 Member Typedef Documentation

result_type

```
template<typename _RealType = double>
typedef _RealType std::normal_distribution< _RealType >::result_type
```

The type of the range of the distribution.

5.701.3 Constructor & Destructor Documentation

normal_distribution()

```
template<typename _RealType = double>
std::normal_distribution< _RealType >::normal_distribution (
            result_type __mean,
            result_type __stddev = result_type(1))  [inline], [explicit]
```

Constructs a normal distribution with parameters $mean$ and standard deviation.

5.701.4 Member Function Documentation

max()

```
template<typename _RealType = double>
result_type std::normal_distribution< _RealType >::max () const  [inline]
```

Returns the least upper bound value of the distribution.

mean()

```
template<typename _RealType = double>
_RealType std::normal_distribution< _RealType >::mean () const  [inline]
```

Returns the mean of the distribution.

min()

```
template<typename _RealType = double>
result_type std::normal_distribution< _RealType >::min () const  [inline]
```

Returns the greatest lower bound value of the distribution.

operator()() [1/2]

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::normal_distribution< _RealType >::operator() (
            _UniformRandomNumberGenerator & __urng)  [inline]
```

Generating functions.
Referenced by operator()().

operator()() [2/2]

```
template<typename _RealType>
template<typename _UniformRandomNumberGenerator>
normal_distribution< _RealType >::result_type std::normal_distribution< _RealType >::operator()
(
            _UniformRandomNumberGenerator & __urng,
            const param_type & __param)
```

Polar method due to Marsaglia.
Devroye, L. Non-Uniform Random Variates Generation. Springer-Verlag, New York, 1986, Ch. V, Sect. 4.4.
References std::log(), and std::sqrt().

param() [1/2]

```
template<typename _RealType = double>
param_type std::normal_distribution< _RealType >::param () const  [inline]
```

Returns the parameter set of the distribution.

param() [2/2]

```
template<typename _RealType = double>
void std::normal_distribution< _RealType >::param (
            const param_type & __param)  [inline]
```

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

reset()

```
template<typename _RealType = double>
void std::normal_distribution< _RealType >::reset ()  [inline]
```

Resets the distribution state.

stddev()

```
template<typename _RealType = double>
_RealType std::normal_distribution< _RealType >::stddev () const  [inline]
```

Returns the standard deviation of the distribution.

5.701.5 Friends And Related Symbol Documentation

operator<<

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::normal_distribution< _RealType1 > & __x)  [friend]
```

Inserts a normal_distribution random number distribution __x into the output stream __os.

Parameters

__os	An output stream.
__x	A normal_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

operator==

```
template<typename _RealType = double>
template<typename _RealType1>
bool operator== (
            const std::normal_distribution< _RealType1 > & __d1,
            const std::normal_distribution< _RealType1 > & __d2)  [friend]
```

Return true if two normal distributions have the same parameters and the sequences that would be generated are equal.

operator>>

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            std::normal_distribution< _RealType1 > & __x)  [friend]
```

Extracts a normal_distribution random number distribution __x from the input stream __is.

Parameters

__is	An input stream.
__x	A normal_distribution random number generator engine.

Returns

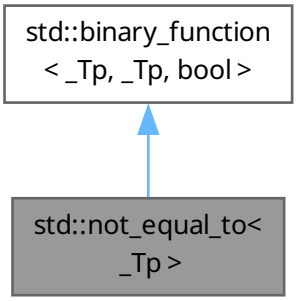
The input stream with __x extracted or in an error state.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

5.702 std::not_equal_to< _Tp > Struct Template Reference

`#include <stl_function.h>`
Inheritance diagram for std::not_equal_to< _Tp >:



Public Types

- typedef _Tp first_argument_type
- typedef bool result_type
- typedef _Tp second_argument_type

Public Member Functions

- constexpr bool **operator()** (const _Tp &__x, const _Tp &__y) const

5.702.1 Detailed Description

template<typename _Tp>
struct std::not_equal_to< _Tp >

One of the comparison functors.

5.702.2 Member Typedef Documentation

first_argument_type

`typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type [inherited]`
`first_argument_type` is the type of the first argument

result_type

typedef bool std::binary_function< _Tp, _Tp, bool >::result_type [inherited]
result_type is the return type

second_argument_type

typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type [inherited]
second_argument_type is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

5.703 std::ranges::not_equal_to Struct Reference

#include <ranges_cmp.h>

Public Types

- using **is_transparent**

Public Member Functions

- template<typename _Tp, typename _Up>
  requires equality_comparable_with<_Tp, _Up>
  constexpr bool **operator()** (_Tp &&__t, _Up &&__u) const noexcept(noexcept(std::declval< _Tp >()==std::declval< _Up >()))

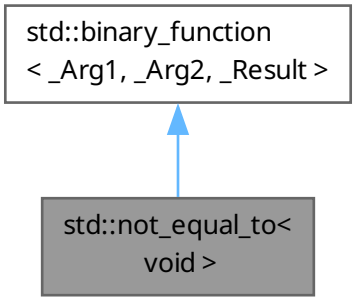
5.703.1 Detailed Description

ranges::not_equal_to function object type.
The documentation for this struct was generated from the following file:

- ranges_cmp.h

5.704 std::not_equal_to< void > Struct Reference

#include <stl_function.h>
Inheritance diagram for std::not_equal_to< void >:



Public Types

- typedef void first_argument_type
- typedef __is_transparent **is_transparent**
- typedef bool result_type
- typedef void second_argument_type

Public Member Functions

- template<typename _Tp, typename _Up>
  constexpr auto **operator()** (_Tp &&__t, _Up &&__u) const noexcept(noexcept(std::forward< _Tp >(__t) !=std::forward< _Up >(__u))) -> decltype(std::forward< _Tp >(__t) !=std::forward< _Up >(__u))
- constexpr bool **operator()** (const void &__x, const void &__y) const

5.704.1 Detailed Description

One of the comparison functors.

5.704.2 Member Typedef Documentation

first_argument_type

```
typedef void std::binary_function< void, void, bool >::first_argument_type
```

`first_argument_type` is the type of the first argument

result_type

```
typedef bool std::binary_function< void, void, bool >::result_type
```

`result_type` is the return type

second_argument_type

```
typedef void std::binary_function< void, void, bool >::second_argument_type
```

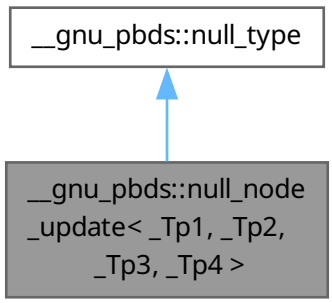
`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

5.705 __gnu_pbds::null_node_update< _Tp1, _Tp2, _Tp3, _Tp4 > Struct Template Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for __gnu_pbds::null_node_update< _Tp1, _Tp2, _Tp3, _Tp4 >:



5.705.1 Detailed Description

template<typename _Tp1, typename _Tp2, typename _Tp3, typename _Tp4>
struct __gnu_pbds::null_node_update< _Tp1, _Tp2, _Tp3, _Tp4 >

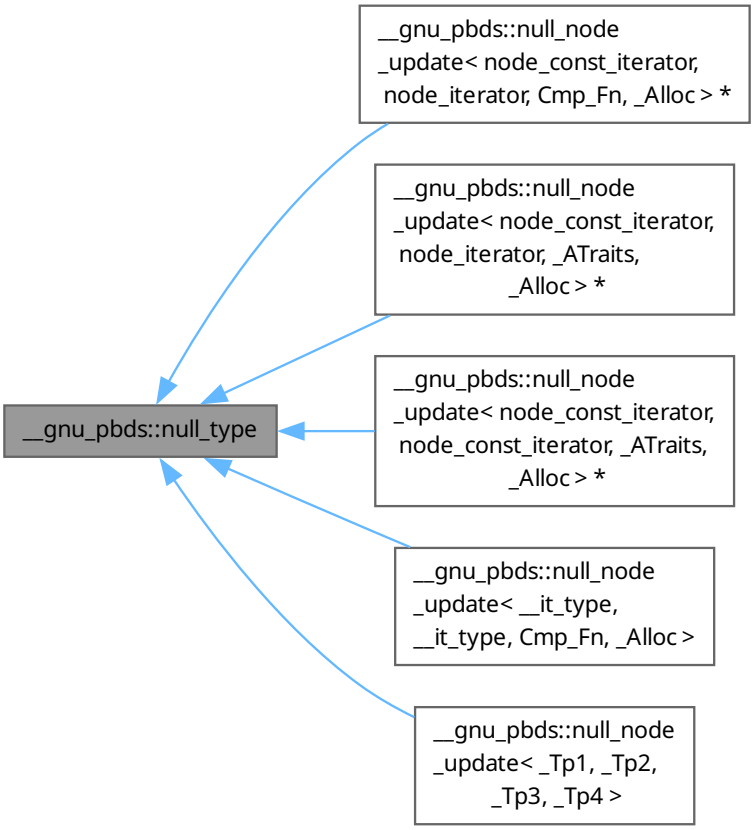
A null node updator, indicating that no node updates are required.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

5.706 __gnu_pbds::null_type Struct Reference

`#include <tag_and_trait.hpp>`

Inheritance diagram for __gnu_pbds::null_type:



5.706.1 Detailed Description

Represents no type, or absence of type, for template tricks.
In a mapped-policy, indicates that an associative container is a set.
In a list-update policy, indicates that each link does not need metadata.
In a hash policy, indicates that the combining hash function is actually a ranged hash function.
In a probe policy, indicates that the combining probe function is actually a ranged probe function.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

5.707 std::experimental::fundamentals_v1::nullopt_t Struct Reference

`#include <optional>`

Public Types

- enum class **_Construct** { **_Token** }

Public Member Functions

- constexpr **nullopt_t** (_Construct)

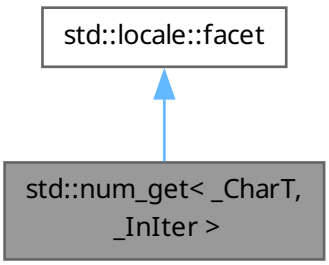
5.707.1 Detailed Description

Tag type to disengage optional objects.
The documentation for this struct was generated from the following file:

- experimental/optional

5.708 std::num_get< _CharT, _InIter > Class Template Reference

`#include <locale_facets.h>`
Inheritance diagram for std::num_get< _CharT, _InIter >:



Public Types

- typedef _CharT char_type
- typedef _InIter iter_type

Public Member Functions

- num_get (size_t __refs=0)
- template<typename _ValueT>
  _InIter **_M_extract_int** (_InIter __beg, _InIter __end, ios_base &__io, ios_base::iostate &__err, _ValueT &__v) const
- iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, bool &__v) const
- iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, void *&__v) const
- iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, long &__v) const
- iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned short &__v) const
- iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned int &__v) const
- iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned long &__v) const
- iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, long long &__v) const

- iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned long long &__v) const

- iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, float &__v) const
- iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, double &__v) const
- iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, long double &__v) const

Static Public Attributes

- static locale::id id

Protected Member Functions

- virtual ~num_get ()
- iter_type **_M_extract_float** (iter_type, iter_type, ios_base &, ios_base::iostate &, string &) const
- template<typename _ValueT>
  iter_type **_M_extract_int** (iter_type, iter_type, ios_base &, ios_base::iostate &, _ValueT &) const
- template<typename _CharT2>
  __gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, int >::__type **_M_find** (const _CharT2 *, size_t _len, _CharT2 __c) const
- template<typename _CharT2>
  __gnu_cxx::__enable_if<!__is_char< _CharT2 >::__value, int >::__type **_M_find** (const _CharT2 *__zero, size_t __len, _CharT2 __c) const

- virtual iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, bool &) const
- virtual iter_type do_get (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, long &__v) const
- virtual iter_type do_get (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned short &__v) const
- virtual iter_type do_get (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned int &__v) const
- virtual iter_type do_get (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned long &__v) const
- virtual iter_type do_get (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, long long &__v) const
- virtual iter_type do_get (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned long long &__v) const
- virtual iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, float &) const
- virtual iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, double &) const
- virtual iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, long double &) const
- virtual iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, void *&) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

5.708.1 Detailed Description

template<typename _CharT, typename _InIter>
class std::num_get< _CharT, _InIter >

Primary class template num_get.
This facet encapsulates the code to parse and return a number from a string. It is used by the istream numeric extraction operators.
The num_get template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the num_get facet.

5.708.2 Member Typedef Documentation

char_type

```
template<typename _CharT, typename _InIter>
typedef _CharT std::num_get< _CharT, _InIter >::char_type
```

Public typedefs.

iter_type

```
template<typename _CharT, typename _InIter>
typedef _InIter std::num_get< _CharT, _InIter >::iter_type
```

Public typedefs.

5.708.3 Constructor & Destructor Documentation

num_get()

```
template<typename _CharT, typename _InIter>
std::num_get< _CharT, _InIter >::num_get (
            size_t __refs = 0)  [inline], [explicit]
```

Constructor performs initialization.
This is the constructor provided by the standard.

Parameters

__refs	Passed to the base facet class.

~num_get()

```
template<typename _CharT, typename _InIter>
virtual std::num_get< _CharT, _InIter >::~num_get ()  [inline], [protected], [virtual]
```

Destructor.

5.708.4 Member Function Documentation

do_get() [1/11]

```
template<typename _CharT, typename _InIter>
virtual iter_type std::num_get< _CharT, _InIter >::do_get (
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
```

```
            ios_base::iostate & __err,
            long & __v) const  [inline], [protected], [virtual]
```

Numeric parsing.
Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

get() for more details.

Parameters

__beg	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

do_get() [2/11]

```
template<typename _CharT, typename _InIter>
virtual iter_type std::num_get< _CharT, _InIter >::do_get (
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            long long & __v) const  [inline], [protected], [virtual]
```

Numeric parsing.
Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

get() for more details.

Parameters

__beg	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

do_get() [3/11]

```
template<typename _CharT, typename _InIter>
virtual iter_type std::num_get< _CharT, _InIter >::do_get (
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            unsigned int & __v) const  [inline], [protected], [virtual]
```

Numeric parsing.
Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

get() for more details.

Parameters

__beg	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

do_get() [4/11]

```
template<typename _CharT, typename _InIter>
virtual iter_type std::num_get< _CharT, _InIter >::do_get (
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            unsigned long & __v) const  [inline], [protected], [virtual]
```

Numeric parsing.
Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

get() for more details.

Parameters

__beg	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

do_get() [5/11]

```
template<typename _CharT, typename _InIter>
virtual iter_type std::num_get< _CharT, _InIter >::do_get (
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            unsigned long long & __v) const  [inline], [protected], [virtual]
```

Numeric parsing.
Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

get() for more details.

Parameters

__beg	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

do_get() [6/11]

```
template<typename _CharT, typename _InIter>
virtual iter_type std::num_get< _CharT, _InIter >::do_get (
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            unsigned short & __v) const  [inline], [protected], [virtual]
```

Numeric parsing.
Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

get() for more details.

Parameters

__beg	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.

__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

do_get() [7/11]

```
template<typename _CharT, typename _InIter>
_InIter std::num_get< _CharT, _InIter >::do_get (
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            bool & __v) const  [protected], [virtual]
```

Numeric parsing.
Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

get() for more details.

Parameters

__beg	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

References std::ios_base::_M_getloc(), std::ios_base::boolalpha, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::flags(), and std::ios_base::goodbit.
Referenced by std::num_get< _CharT, istreambuf_iterator< _CharT, _Traits > >::get(), std::num_get< _CharT, istreambuf_iterator< _CharT, _Traits > >::get(), std::num_get< _CharT, istreambuf_iterator< _CharT, _Traits > >::get(), std::num_get< _CharT, istreambuf_iterator< _CharT, _Traits > >::get(), std::num_get< _CharT, istreambuf_iterator< _CharT, _Traits > >::get(), std::num_get< _CharT, istreambuf_iterator< _CharT, _Traits > >::get(), std::num_get< _CharT, istreambuf_iterator< _CharT, _Traits > >::get(), and std::num_get< _CharT, istreambuf_iterator< _CharT, _Traits > >::get().

do_get() [8/11]

```
template<typename _CharT, typename _InIter>
_InIter std::num_get< _CharT, _InIter >::do_get (
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
```

```
            ios_base::iostate & __err,
            double & __v) const  [protected], [virtual]
```

Numeric parsing.
Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

get() for more details.

Parameters

__beg	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

References std::basic_string< _CharT, _Traits, _Alloc >::c_str(), std::ios_base::eofbit, and std::basic_string< _CharT, _Traits, _Alloc >::r

do_get() [9/11]

```
template<typename _CharT, typename _InIter>
_InIter std::num_get< _CharT, _InIter >::do_get (
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            float & __v) const  [protected], [virtual]
```

Numeric parsing.
Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

get() for more details.

Parameters

__beg	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

References std::basic_string< _CharT, _Traits, _Alloc >::c_str(), std::ios_base::eofbit, and std::basic_string< _CharT, _Traits, _Alloc >::r

do_get() [10/11]

```
template<typename _CharT, typename _InIter>
_InIter std::num_get< _CharT, _InIter >::do_get (
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            long double & __v) const  [protected], [virtual]
```

Numeric parsing.
Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

get() for more details.

Parameters

__beg	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

References std::basic_string< _CharT, _Traits, _Alloc >::c_str(), std::ios_base::eofbit, and std::basic_string< _CharT, _Traits, _Alloc >::r

do_get() [11/11]

```
template<typename _CharT, typename _InIter>
_InIter std::num_get< _CharT, _InIter >::do_get (
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            void *& __v) const  [protected], [virtual]
```

Numeric parsing.
Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

get() for more details.

Parameters

__beg	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

References std::ios_base::basefield, std::ios_base::flags(), std::ios_base::fmtflags, and std::ios_base::hex.

get() [1/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
            iter_type __in,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            bool & __v) const  [inline]
```

Numeric parsing.
Parses the input stream into the bool *v*. It does so by calling num_get::do_get().
If ios_base::boolalpha is set, attempts to read ctype<CharT>::truename() or ctype<CharT>::falsename(). Sets *v* to true or false if successful. Sets err to ios_base::failbit if reading the string fails. Sets err to ios_base::eofbit if the stream is emptied.
If ios_base::boolalpha is not set, proceeds as with reading a long, except if the value is 1, sets *v* to true, if the value is 0, sets *v* to false, and otherwise set err to ios_base::failbit.

Parameters

__in	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

Referenced by std::basic_istream< _CharT, _Traits >::operator>>(), and std::basic_istream< _CharT, _Traits >::operator>>().

get() [2/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
            iter_type __in,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            double & __v) const  [inline]
```

Numeric parsing.
Parses the input stream into the integral variable *v*. It does so by calling num_get::do_get().
The input characters are parsed like the scanf g specifier. The matching type length modifier is also used.
The decimal point character used is numpunct::decimal_point(). Digit grouping is interpreted according to numpunct↩ ::grouping() and numpunct::thousands_sep(). If the pattern of digit groups isn't consistent, sets err to ios_base::failbit.
If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios_base::failbit and leaves *v* unaltered. Sets err to ios_base::eofbit if the stream is emptied.

Parameters

__in	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

get() [3/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
            iter_type __in,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            float & __v) const  [inline]
```

Numeric parsing.
Parses the input stream into the integral variable *v*. It does so by calling num_get::do_get().
The input characters are parsed like the scanf g specifier. The matching type length modifier is also used.
The decimal point character used is numpunct::decimal_point(). Digit grouping is interpreted according to numpunct↩ ::grouping() and numpunct::thousands_sep(). If the pattern of digit groups isn't consistent, sets err to ios_base::failbit.
If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios_base::failbit and leaves *v* unaltered.
Sets err to ios_base::eofbit if the stream is emptied.

Parameters

__in	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

get() [4/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
            iter_type __in,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            long & __v) const  [inline]
```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling num_get::do_get().
Parsing is affected by the flag settings in *io*.
The basic parse is affected by the value of io.flags() & ios_base::basefield. If equal to ios_base::oct, parses like the scanf o specifier. Else if equal to ios_base::hex, parses like X specifier. Else if basefield equal to 0, parses like the i specifier. Otherwise, parses like d for signed and u for unsigned types. The matching type length modifier is also used.
Digit grouping is interpreted according to numpunct::grouping() and numpunct::thousands_sep(). If the pattern of digit groups isn't consistent, sets err to ios_base::failbit.
If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios_base::failbit and leaves *v* unaltered. Sets err to ios_base::eofbit if the stream is emptied.

Parameters

__in	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

get() [5/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
             iter_type __in,
             iter_type __end,
             ios_base & __io,
             ios_base::iostate & __err,
             long double & __v) const  [inline]
```

Numeric parsing.
Parses the input stream into the integral variable *v*. It does so by calling num_get::do_get().
The input characters are parsed like the scanf g specifier. The matching type length modifier is also used.
The decimal point character used is numpunct::decimal_point(). Digit grouping is interpreted according to numpunct↩ ::grouping() and numpunct::thousands_sep(). If the pattern of digit groups isn't consistent, sets err to ios_base::failbit.
If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios_base::failbit and leaves *v* unaltered. Sets err to ios_base::eofbit if the stream is emptied.

Parameters

__in	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

get() [6/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
            iter_type __in,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            long long & __v) const  [inline]
```

Numeric parsing.
Parses the input stream into the integral variable *v*. It does so by calling num_get::do_get().
Parsing is affected by the flag settings in *io*.
The basic parse is affected by the value of io.flags() & ios_base::basefield. If equal to ios_base::oct, parses like the scanf o specifier. Else if equal to ios_base::hex, parses like X specifier. Else if basefield equal to 0, parses like the i specifier. Otherwise, parses like d for signed and u for unsigned types. The matching type length modifier is also used.
Digit grouping is interpreted according to numpunct::grouping() and numpunct::thousands_sep(). If the pattern of digit groups isn't consistent, sets err to ios_base::failbit.
If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios_base::failbit and leaves *v* unaltered. Sets err to ios_base::eofbit if the stream is emptied.

Parameters

__in	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

get() [7/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
            iter_type __in,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            unsigned int & __v) const  [inline]
```

Numeric parsing.
Parses the input stream into the integral variable *v*. It does so by calling num_get::do_get().
Parsing is affected by the flag settings in *io*.
The basic parse is affected by the value of io.flags() & ios_base::basefield. If equal to ios_base::oct, parses like the scanf o specifier. Else if equal to ios_base::hex, parses like X specifier. Else if basefield equal to 0, parses like the i specifier. Otherwise, parses like d for signed and u for unsigned types. The matching type length modifier is also used.
Digit grouping is interpreted according to numpunct::grouping() and numpunct::thousands_sep(). If the pattern of digit groups isn't consistent, sets err to ios_base::failbit.
If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios_base::failbit and leaves *v* unaltered. Sets err to ios_base::eofbit if the stream is emptied.

Parameters

__in	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

get() [8/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
            iter_type __in,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            unsigned long & __v) const  [inline]
```

Numeric parsing.
Parses the input stream into the integral variable *v*. It does so by calling num_get::do_get().
Parsing is affected by the flag settings in *io*.
The basic parse is affected by the value of io.flags() & ios_base::basefield. If equal to ios_base::oct, parses like the scanf o specifier. Else if equal to ios_base::hex, parses like X specifier. Else if basefield equal to 0, parses like the i specifier. Otherwise, parses like d for signed and u for unsigned types. The matching type length modifier is also used.
Digit grouping is interpreted according to numpunct::grouping() and numpunct::thousands_sep(). If the pattern of digit groups isn't consistent, sets err to ios_base::failbit.
If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios_base::failbit and leaves *v* unaltered. Sets err to ios_base::eofbit if the stream is emptied.

Parameters

__in	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

get() [9/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
            iter_type __in,
            iter_type __end,
            ios_base & __io,
```

```
            ios_base::iostate & __err,
            unsigned long long & __v) const  [inline]
```

Numeric parsing.
Parses the input stream into the integral variable *v*. It does so by calling num_get::do_get().
Parsing is affected by the flag settings in *io*.
The basic parse is affected by the value of io.flags() & ios_base::basefield. If equal to ios_base::oct, parses like the scanf o specifier. Else if equal to ios_base::hex, parses like X specifier. Else if basefield equal to 0, parses like the i specifier. Otherwise, parses like d for signed and u for unsigned types. The matching type length modifier is also used.
Digit grouping is interpreted according to numpunct::grouping() and numpunct::thousands_sep(). If the pattern of digit groups isn't consistent, sets err to ios_base::failbit.
If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios_base::failbit and leaves *v* unaltered. Sets err to ios_base::eofbit if the stream is emptied.

Parameters

__in	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

get() [10/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
            iter_type __in,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            unsigned short & __v) const  [inline]
```

Numeric parsing.
Parses the input stream into the integral variable *v*. It does so by calling num_get::do_get().
Parsing is affected by the flag settings in *io*.
The basic parse is affected by the value of io.flags() & ios_base::basefield. If equal to ios_base::oct, parses like the scanf o specifier. Else if equal to ios_base::hex, parses like X specifier. Else if basefield equal to 0, parses like the i specifier. Otherwise, parses like d for signed and u for unsigned types. The matching type length modifier is also used.
Digit grouping is interpreted according to numpunct::grouping() and numpunct::thousands_sep(). If the pattern of digit groups isn't consistent, sets err to ios_base::failbit.
If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios_base::failbit and leaves *v* unaltered. Sets err to ios_base::eofbit if the stream is emptied.

Parameters

__in	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

get() [11/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
            iter_type __in,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            void *& __v) const  [inline]
```

Numeric parsing.
Parses the input stream into the pointer variable *v*. It does so by calling num_get::do_get().
The input characters are parsed like the scanf p specifier.
Digit grouping is interpreted according to numpunct::grouping() and numpunct::thousands_sep(). If the pattern of digit groups isn't consistent, sets err to ios_base::failbit.
Note that the digit grouping effect for pointers is a bit ambiguous in the standard and shouldn't be relied on. See DR 344.
If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios_base::failbit and leaves *v* unaltered. Sets err to ios_base::eofbit if the stream is emptied.

Parameters

__in	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

5.708.5 Member Data Documentation

id

```
template<typename _CharT, typename _InIter>
locale::id std::num_get< _CharT, _InIter >::id  [static]
```

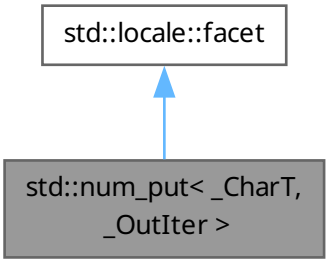
Numpunct facet id.
The documentation for this class was generated from the following files:

- locale_facets.h
- locale_facets.tcc

5.709 std::num_put< _CharT, _OutIter > Class Template Reference

```
#include <locale_facets.h>
```

Inheritance diagram for std::num_put< _CharT, _OutIter >:



Public Types

- typedef _CharT char_type
- typedef _OutIter iter_type

Public Member Functions

- num_put (size_t __refs=0)
- template<typename _ValueT>
  _OutIter **_M_insert_float** (_OutIter __s, ios_base &__io, _CharT __fill, char __mod, _ValueT __v) const
- template<typename _ValueT>
  _OutIter **_M_insert_int** (_OutIter __s, ios_base &__io, _CharT __fill, _ValueT __v) const
- iter_type put (iter_type __s, ios_base &__io, char_type __fill, bool __v) const
- iter_type put (iter_type __s, ios_base &__io, char_type __fill, const void *__v) const
- iter_type put (iter_type __s, ios_base &__io, char_type __fill, long __v) const
- iter_type put (iter_type __s, ios_base &__io, char_type __fill, unsigned long __v) const
- iter_type put (iter_type __s, ios_base &__io, char_type __fill, long long __v) const
- iter_type put (iter_type __s, ios_base &__io, char_type __fill, unsigned long long __v) const
- iter_type put (iter_type __s, ios_base &__io, char_type __fill, double __v) const
- iter_type put (iter_type __s, ios_base &__io, char_type __fill, long double __v) const

Static Public Attributes

- static locale::id id

Protected Member Functions

- virtual ~num_put ()
- void **_M_group_float** (const char *__grouping, size_t __grouping_size, char_type __sep, const char_type *__p, char_type *__new, char_type *__cs, int &__len) const

- void **_M_group_int** (const char *__grouping, size_t __grouping_size, char_type __sep, ios_base &__io, char_type *__new, char_type *__cs, int &__len) const
- template<typename _ValueT>
  iter_type **_M_insert_float** (iter_type, ios_base &__io, char_type __fill, char __mod, _ValueT __v) const
- template<typename _ValueT>
  iter_type **_M_insert_int** (iter_type, ios_base &__io, char_type __fill, _ValueT __v) const
- void **_M_pad** (char_type __fill, streamsize __w, ios_base &__io, char_type *__new, const char_type *__cs, int &__len) const

- virtual iter_type do_put (iter_type __s, ios_base &__io, char_type __fill, bool __v) const
- virtual iter_type do_put (iter_type __s, ios_base &__io, char_type __fill, long __v) const
- virtual iter_type do_put (iter_type __s, ios_base &__io, char_type __fill, unsigned long __v) const
- virtual iter_type do_put (iter_type __s, ios_base &__io, char_type __fill, long long __v) const
- virtual iter_type do_put (iter_type __s, ios_base &__io, char_type __fill, unsigned long long __v) const
- virtual iter_type do_put (iter_type, ios_base &, char_type, double) const
- virtual iter_type do_put (iter_type, ios_base &, char_type, long double) const
- virtual iter_type do_put (iter_type, ios_base &, char_type, const void *) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

5.709.1 Detailed Description

template<typename _CharT, typename _OutIter>
class std::num_put< _CharT, _OutIter >

Primary class template num_put.
This facet encapsulates the code to convert a number to a string. It is used by the ostream numeric insertion operators. The num_put template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the num_put facet.

5.709.2 Member Typedef Documentation

char_type

```
template<typename _CharT, typename _OutIter>
typedef _CharT std::num_put< _CharT, _OutIter >::char_type
```

Public typedefs.

iter_type

```
template<typename _CharT, typename _OutIter>
typedef _OutIter std::num_put< _CharT, _OutIter >::iter_type
```

Public typedefs.

5.709.3 Constructor & Destructor Documentation

num_put()

```
template<typename _CharT, typename _OutIter>
std::num_put< _CharT, _OutIter >::num_put (
            size_t __refs = 0)  [inline], [explicit]
```

Constructor performs initialization.
This is the constructor provided by the standard.

Parameters

__refs	Passed to the base facet class.

~num_put()

```
template<typename _CharT, typename _OutIter>
virtual std::num_put< _CharT, _OutIter >::~num_put ()  [inline], [protected], [virtual]
```

Destructor.

5.709.4 Member Function Documentation

do_put() [1/8]

```
template<typename _CharT, typename _OutIter>
_OutIter std::num_put< _CharT, _OutIter >::do_put (
            iter_type __s,
            ios_base & __io,
            char_type __fill,
            bool __v) const  [protected], [virtual]
```

Numeric formatting.
These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

_↩ _s	Stream to write to.
_↩ _io	Source of locale and flags.
_↩ _fill	Char_type to use for filling.
_↩ _v	Value to format and insert.

Returns

Iterator after writing.

References std::ios_base::_M_getloc(), std::ios_base::adjustfield, std::ios_base::boolalpha, std::ios_base::flags(), std::ios_base::fmtflags, std::ios_base::left, and std::ios_base::width().
Referenced by std::num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > >::put(), std::num_put< _CharT, ostreambuf_iterator< _C
std::num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > >::put(), std::num_put< _CharT, ostreambuf_iterator< _CharT, _Traits >
std::num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > >::put(), std::num_put< _CharT, ostreambuf_iterator< _CharT, _Traits >
std::num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > >::put(), and std::num_put< _CharT, ostreambuf_iterator< _CharT, _Tra

do_put() [2/8]

```
template<typename _CharT, typename _OutIter>
virtual iter_type std::num_put< _CharT, _OutIter >::do_put (
            iter_type __s,
            ios_base & __io,
            char_type __fill,
            long __v) const  [inline], [protected], [virtual]
```

Numeric formatting.
These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

_↩ _s	Stream to write to.
_↩ _io	Source of locale and flags.
_↩ _fill	Char_type to use for filling.
_↩ _v	Value to format and insert.

Returns

Iterator after writing.

do_put() [3/8]

```
template<typename _CharT, typename _OutIter>
virtual iter_type std::num_put< _CharT, _OutIter >::do_put (
            iter_type __s,
            ios_base & __io,
            char_type __fill,
            long long __v) const  [inline], [protected], [virtual]
```

Numeric formatting.
These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

_↩ _s	Stream to write to.
_↩ _io	Source of locale and flags.
_↩ _fill	Char_type to use for filling.
_↩ _v	Value to format and insert.

Returns

Iterator after writing.

do_put() [4/8]

```
template<typename _CharT, typename _OutIter>
virtual iter_type std::num_put< _CharT, _OutIter >::do_put (
            iter_type __s,
            ios_base & __io,
            char_type __fill,
            unsigned long __v) const  [inline], [protected], [virtual]
```

Numeric formatting.
These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

_↩ _s	Stream to write to.
_↩ _io	Source of locale and flags.
_↩ _fill	Char_type to use for filling.
_↩ _v	Value to format and insert.

Returns

Iterator after writing.

do_put() [5/8]

```
template<typename _CharT, typename _OutIter>
virtual iter_type std::num_put< _CharT, _OutIter >::do_put (
            iter_type __s,
            ios_base & __io,
            char_type __fill,
            unsigned long long __v) const  [inline], [protected], [virtual]
```

Numeric formatting.
These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

_↩ _s	Stream to write to.
_↩ _io	Source of locale and flags.
_↩ _fill	Char_type to use for filling.
_↩ _v	Value to format and insert.

Returns

Iterator after writing.

do_put() [6/8]

```
template<typename _CharT, typename _OutIter>
_OutIter std::num_put< _CharT, _OutIter >::do_put (
            iter_type __s,
            ios_base & __io,
            char_type __fill,
            const void * __v) const  [protected], [virtual]
```

Numeric formatting.
These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

_↩ _s	Stream to write to.
_↩ _io	Source of locale and flags.
_↩ _fill	Char_type to use for filling.
_↩ _v	Value to format and insert.

Returns

Iterator after writing.

References std::ios_base::basefield, std::ios_base::flags(), std::ios_base::fmtflags, std::ios_base::hex, std::ios_base::showbase, and std::ios_base::uppercase.

do_put() [7/8]

```
template<typename _CharT, typename _OutIter>
_OutIter std::num_put< _CharT, _OutIter >::do_put (
            iter_type __s,
            ios_base & __io,
            char_type __fill,
            double __v) const  [protected], [virtual]
```

Numeric formatting.
These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

_↩ _s	Stream to write to.
_↩ _io	Source of locale and flags.
_↩ _fill	Char_type to use for filling.
_↩ _v	Value to format and insert.

Returns

Iterator after writing.

do_put() [8/8]

```
template<typename _CharT, typename _OutIter>
_OutIter std::num_put< _CharT, _OutIter >::do_put (
            iter_type __s,
            ios_base & __io,
            char_type __fill,
            long double __v) const  [protected], [virtual]
```

Numeric formatting.
These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

_↩ _s	Stream to write to.
_↩ _io	Source of locale and flags.
_↩ _fill	Char_type to use for filling.
_↩ _v	Value to format and insert.

Returns

Iterator after writing.

put() [1/8]

```
template<typename _CharT, typename _OutIter>
iter_type std::num_put< _CharT, _OutIter >::put (
            iter_type __s,
            ios_base & __io,
            char_type __fill,
            bool __v) const  [inline]
```

Numeric formatting.
Formats the boolean *v* and inserts it into a stream. It does so by calling num_put::do_put().
If ios_base::boolalpha is set, writes ctype<CharT>::truename() or ctype<CharT>::falsename(). Otherwise formats *v* as an int.

Parameters

_↩ _s	Stream to write to.
_↩ _io	Source of locale and flags.
_↩ _fill	Char_type to use for filling.
_↩ _v	Value to format and insert.

Returns

Iterator after writing.

Referenced by std::basic_ostream< _CharT, _Traits >::operator<<().

put() [2/8]

```
template<typename _CharT, typename _OutIter>
iter_type std::num_put< _CharT, _OutIter >::put (
            iter_type __s,
            ios_base & __io,
            char_type __fill,
            const void * __v) const  [inline]
```

Numeric formatting.
Formats the pointer value *v* and inserts it into a stream. It does so by calling num_put::do_put().
This function formats *v* as an unsigned long with ios_base::hex and ios_base::showbase set.

Parameters

_↩ _*s*	Stream to write to.
_↩ _*io*	Source of locale and flags.
_↩ _*fill*	Char_type to use for filling.
_↩ _*v*	Value to format and insert.

Returns

Iterator after writing.

put() [3/8]

```
template<typename _CharT, typename _OutIter>
iter_type std::num_put< _CharT, _OutIter >::put (
            iter_type __s,
            ios_base & __io,
            char_type __fill,
            double __v) const  [inline]
```

Numeric formatting.
Formats the floating point value *v* and inserts it into a stream. It does so by calling num_put::do_put().
Formatting is affected by the flag settings in *io*.
The basic format is affected by the value of io.flags() & ios_base::floatfield. If equal to ios_base::fixed, formats like the printf f specifier. Else if equal to ios_base::scientific, formats like e or E with ios_base::uppercase unset or set respectively. Otherwise, formats like g or G depending on uppercase. Note that if both fixed and scientific are set, the effect will also be like g or G.
The output precision is given by io.precision(). This precision is capped at numeric_limits::digits10 + 2 (different for double and long double). The default precision is 6.
If ios_base::showpos is set, '+' is output before positive values. If ios_base::showpoint is set, a decimal point will always be output.

The decimal point character used is numpunct::decimal_point(). Thousands separators are inserted according to numpunct::grouping() and numpunct::thousands_sep().
If io.width() is non-zero, enough *fill* characters are inserted to make the result at least that wide. If (io.flags() & ios_↩ base::adjustfield) == ios_base::left, result is padded at the end. If ios_base::internal, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

Parameters

_↩ _*s*	Stream to write to.
_↩ _*io*	Source of locale and flags.
_↩ _*fill*	Char_type to use for filling.
_↩ _*v*	Value to format and insert.

Returns

Iterator after writing.

put() [4/8]

```
template<typename _CharT, typename _OutIter>
iter_type std::num_put< _CharT, _OutIter >::put (
            iter_type __s,
            ios_base & __io,
            char_type __fill,
            long __v) const  [inline]
```

Numeric formatting.
Formats the integral value *v* and inserts it into a stream. It does so by calling num_put::do_put().
Formatting is affected by the flag settings in *io*.
The basic format is affected by the value of io.flags() & ios_base::basefield. If equal to ios_base::oct, formats like the printf o specifier. Else if equal to ios_base::hex, formats like x or X with ios_base::uppercase unset or set respectively. Otherwise, formats like d, ld, lld for signed and u, lu, llu for unsigned values. Note that if both oct and hex are set, neither will take effect.
If ios_base::showpos is set, '+' is output before positive values. If ios_base::showbase is set, '0' precedes octal values (except 0) and '0[xX]' precedes hex values.
The decimal point character used is numpunct::decimal_point(). Thousands separators are inserted according to numpunct::grouping() and numpunct::thousands_sep().
If io.width() is non-zero, enough *fill* characters are inserted to make the result at least that wide. If (io.flags() & ios_↩ base::adjustfield) == ios_base::left, result is padded at the end. If ios_base::internal, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

Parameters

_↩ _*s*	Stream to write to.
_↩ _*io*	Source of locale and flags.
_↩ _*fill*	Char_type to use for filling.

_↩ _v	Value to format and insert.

Returns

Iterator after writing.

put() [5/8]

```
template<typename _CharT, typename _OutIter>
iter_type std::num_put< _CharT, _OutIter >::put (
            iter_type __s,
            ios_base & __io,
            char_type __fill,
            long double __v) const  [inline]
```

Numeric formatting.
Formats the floating point value *v* and inserts it into a stream. It does so by calling num_put::do_put().
Formatting is affected by the flag settings in *io*.
The basic format is affected by the value of io.flags() & ios_base::floatfield. If equal to ios_base::fixed, formats like the printf f specifier. Else if equal to ios_base::scientific, formats like e or E with ios_base::uppercase unset or set respectively. Otherwise, formats like g or G depending on uppercase. Note that if both fixed and scientific are set, the effect will also be like g or G.
The output precision is given by io.precision(). This precision is capped at numeric_limits::digits10 + 2 (different for double and long double). The default precision is 6.
If ios_base::showpos is set, '+' is output before positive values. If ios_base::showpoint is set, a decimal point will always be output.
The decimal point character used is numpunct::decimal_point(). Thousands separators are inserted according to numpunct::grouping() and numpunct::thousands_sep().
If io.width() is non-zero, enough *fill* characters are inserted to make the result at least that wide. If (io.flags() & ios_↩ base::adjustfield) == ios_base::left, result is padded at the end. If ios_base::internal, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

Parameters

_↩ _s	Stream to write to.
_↩ _io	Source of locale and flags.
_↩ _fill	Char_type to use for filling.
_↩ _v	Value to format and insert.

Returns

Iterator after writing.

put() [6/8]

```
template<typename _CharT, typename _OutIter>
iter_type std::num_put< _CharT, _OutIter >::put (
```

```
            iter_type __s,
            ios_base & __io,
            char_type __fill,
            long long __v) const  [inline]
```

Numeric formatting.

Formats the integral value *v* and inserts it into a stream. It does so by calling num_put::do_put().

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of io.flags() & ios_base::basefield. If equal to ios_base::oct, formats like the printf o specifier. Else if equal to ios_base::hex, formats like x or X with ios_base::uppercase unset or set respectively. Otherwise, formats like d, ld, lld for signed and u, lu, llu for unsigned values. Note that if both oct and hex are set, neither will take effect.

If ios_base::showpos is set, '+' is output before positive values. If ios_base::showbase is set, '0' precedes octal values (except 0) and '0[xX]' precedes hex values.

The decimal point character used is numpunct::decimal_point(). Thousands separators are inserted according to numpunct::grouping() and numpunct::thousands_sep().

If io.width() is non-zero, enough *fill* characters are inserted to make the result at least that wide. If (io.flags() & ios_↩ base::adjustfield) == ios_base::left, result is padded at the end. If ios_base::internal, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

Parameters

_↩ _*s*	Stream to write to.
_↩ _*io*	Source of locale and flags.
_↩ _*fill*	Char_type to use for filling.
_↩ _*v*	Value to format and insert.

Returns

Iterator after writing.

put() [7/8]

```
template<typename _CharT, typename _OutIter>
iter_type std::num_put< _CharT, _OutIter >::put (
            iter_type __s,
            ios_base & __io,
            char_type __fill,
            unsigned long __v) const  [inline]
```

Numeric formatting.

Formats the integral value *v* and inserts it into a stream. It does so by calling num_put::do_put().

Formatting is affected by the flag settings in *io*.

The basic format is affected by the value of io.flags() & ios_base::basefield. If equal to ios_base::oct, formats like the printf o specifier. Else if equal to ios_base::hex, formats like x or X with ios_base::uppercase unset or set respectively. Otherwise, formats like d, ld, lld for signed and u, lu, llu for unsigned values. Note that if both oct and hex are set, neither will take effect.

If ios_base::showpos is set, '+' is output before positive values. If ios_base::showbase is set, '0' precedes octal values (except 0) and '0[xX]' precedes hex values.

The decimal point character used is numpunct::decimal_point(). Thousands separators are inserted according to numpunct::grouping() and numpunct::thousands_sep().

If io.width() is non-zero, enough *fill* characters are inserted to make the result at least that wide. If (io.flags() & ios_↩ base::adjustfield) == ios_base::left, result is padded at the end. If ios_base::internal, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

Parameters

_↩ _*s*	Stream to write to.
_↩ _*io*	Source of locale and flags.
_↩ _*fill*	Char_type to use for filling.
_↩ _*v*	Value to format and insert.

Returns

Iterator after writing.

put() [8/8]

```
template<typename _CharT, typename _OutIter>
iter_type std::num_put< _CharT, _OutIter >::put (
             iter_type __s,
             ios_base & __io,
             char_type __fill,
             unsigned long long __v) const  [inline]
```

Numeric formatting.
Formats the integral value *v* and inserts it into a stream. It does so by calling num_put::do_put().
Formatting is affected by the flag settings in *io*.
The basic format is affected by the value of io.flags() & ios_base::basefield. If equal to ios_base::oct, formats like the printf o specifier. Else if equal to ios_base::hex, formats like x or X with ios_base::uppercase unset or set respectively. Otherwise, formats like d, ld, lld for signed and u, lu, llu for unsigned values. Note that if both oct and hex are set, neither will take effect.
If ios_base::showpos is set, '+' is output before positive values. If ios_base::showbase is set, '0' precedes octal values (except 0) and '0[xX]' precedes hex values.
The decimal point character used is numpunct::decimal_point(). Thousands separators are inserted according to numpunct::grouping() and numpunct::thousands_sep().
If io.width() is non-zero, enough *fill* characters are inserted to make the result at least that wide. If (io.flags() & ios_↩ base::adjustfield) == ios_base::left, result is padded at the end. If ios_base::internal, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

Parameters

_↩ _*s*	Stream to write to.
_↩ _*io*	Source of locale and flags.
_↩ _*fill*	Char_type to use for filling.
_↩ _*v*	Value to format and insert.

Returns

Iterator after writing.

5.709.5 Member Data Documentation

id

```
template<typename _CharT, typename _OutIter>
locale::id std::num_put< _CharT, _OutIter >::id  [static]
```

Numpunct facet id.

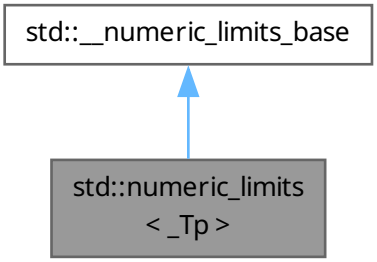
The documentation for this class was generated from the following files:

- locale_facets.h
- locale_facets.tcc

5.710 std::numeric_limits< _Tp > Struct Template Reference

```
#include <limits>
```

Inheritance diagram for std::numeric_limits< _Tp >:



Static Public Member Functions

- static constexpr _Tp denorm_min () noexcept
- static constexpr _Tp epsilon () noexcept
- static constexpr _Tp infinity () noexcept
- static constexpr _Tp lowest () noexcept
- static constexpr _Tp max () noexcept
- static constexpr _Tp min () noexcept
- static constexpr _Tp quiet_NaN () noexcept
- static constexpr _Tp round_error () noexcept
- static constexpr _Tp signaling_NaN () noexcept

Static Public Attributes

- static constexpr int digits
- static constexpr int digits10
- static constexpr float_denorm_style has_denorm
- static constexpr bool has_denorm_loss

- static constexpr bool has_infinity
- static constexpr bool has_quiet_NaN
- static constexpr bool has_signaling_NaN
- static constexpr bool is_bounded
- static constexpr bool is_exact
- static constexpr bool is_iec559
- static constexpr bool is_integer
- static constexpr bool is_modulo
- static constexpr bool is_signed
- static constexpr bool is_specialized
- static constexpr int max_digits10
- static constexpr int max_exponent
- static constexpr int max_exponent10
- static constexpr int min_exponent
- static constexpr int min_exponent10
- static constexpr int radix
- static constexpr float_round_style round_style
- static constexpr bool tinyness_before
- static constexpr bool traps

5.710.1 Detailed Description

template<typename _Tp>
struct std::numeric_limits< _Tp >

Properties of fundamental types.
This class allows a program to obtain information about the representation of a fundamental type on a given platform. For non-fundamental types, the functions will return 0 and the data members will all be `false`.

5.710.2 Member Function Documentation

denorm_min()

```
template<typename _Tp>
static constexpr _Tp std::numeric_limits< _Tp >::denorm_min ()  [inline], [static], [constexpr],
[noexcept]
```

The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

epsilon()

```
template<typename _Tp>
static constexpr _Tp std::numeric_limits< _Tp >::epsilon ()  [inline], [static], [constexpr],
[noexcept]
```

The *machine epsilon:* the difference between 1 and the least value greater than 1 that is representable.
Referenced by std::generate_canonical(), std::binomial_distribution< _IntType >::operator()(), std::poisson_distribution< _IntType >::ope
and std::operator<<().

infinity()

```
template<typename _Tp>
static constexpr _Tp std::numeric_limits< _Tp >::infinity ()  [inline], [static], [constexpr],
[noexcept]
```

The representation of positive infinity, if `has_infinity`.

lowest()

```
template<typename _Tp>
static constexpr _Tp std::numeric_limits< _Tp >::lowest ()  [inline], [static], [constexpr],
[noexcept]
```

A finite value x such that there is no other finite value y where y < x.
Referenced by std::cauchy_distribution< _RealType >::min(), std::extreme_value_distribution< _RealType >::min(), std::normal_distribution< _RealType >::min(), and std::student_t_distribution< _RealType >::min().

max()

```
template<typename _Tp>
static constexpr _Tp std::numeric_limits< _Tp >::max ()  [inline], [static], [constexpr], [noexcept]
```

The maximum finite value.
Referenced by __gnu_parallel::__parallel_random_shuffle_drs(), std::bernoulli_distribution::max(), std::cauchy_distribution< _RealType >
std::chi_squared_distribution< _RealType >::max(), std::exponential_distribution< _RealType >::max(), std::extreme_value_distribution<
std::fisher_f_distribution< _RealType >::max(), std::gamma_distribution< _RealType >::max(), std::geometric_distribution< _IntType >:
std::lognormal_distribution< _RealType >::max(), std::negative_binomial_distribution< _IntType >::max(), std::normal_distribution< _Re
std::poisson_distribution< _IntType >::max(), std::student_t_distribution< _RealType >::max(), std::weibull_distribution< _RealType >::
std::binomial_distribution< _IntType >::operator()(), std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::operat
std::poisson_distribution< _IntType >::operator()(), and std::operator<<().

min()

```
template<typename _Tp>
static constexpr _Tp std::numeric_limits< _Tp >::min ()  [inline], [static], [constexpr], [noexcept]
```

The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.
Referenced by std::bernoulli_distribution::min().

quiet_NaN()

```
template<typename _Tp>
static constexpr _Tp std::numeric_limits< _Tp >::quiet_NaN ()  [inline], [static], [constexpr],
[noexcept]
```

The representation of a quiet Not a Number, if `has_quiet_NaN`.

round_error()

```
template<typename _Tp>
static constexpr _Tp std::numeric_limits< _Tp >::round_error ()  [inline], [static], [constexpr],
[noexcept]
```

The maximum rounding error measurement (see LIA-1).

signaling_NaN()

```
template<typename _Tp>
static constexpr _Tp std::numeric_limits< _Tp >::signaling_NaN ()  [inline], [static], [constexpr],
[noexcept]
```

The representation of a signaling Not a Number, if `has_signaling_NaN`.

5.710.3 Member Data Documentation

digits

```
int std::__numeric_limits_base::digits  [static], [constexpr], [inherited]
```

The number of radix digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of radix digits in the mantissa.
Referenced by std::generate_canonical(), std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::operator()(), std::operator<<(), and std::linear_congruential_engine< uint_fast32_t, 16807UL, 0UL, 2147483647UL >::seed().

digits10

int std::__numeric_limits_base::digits10 [static], [constexpr], [inherited]
The number of base 10 digits that can be represented without change.

has_denorm

float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr], [inherited]
See std::float_denorm_style for more information.

has_denorm_loss

bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr], [inherited]
True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

has_infinity

bool std::__numeric_limits_base::has_infinity [static], [constexpr], [inherited]
True if the type has a representation for positive infinity.

has_quiet_NaN

bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr], [inherited]
True if the type has a representation for a quiet (non-signaling) Not a Number.

has_signaling_NaN

bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr], [inherited]
True if the type has a representation for a signaling Not a Number.

is_bounded

bool std::__numeric_limits_base::is_bounded [static], [constexpr], [inherited]
True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

is_exact

bool std::__numeric_limits_base::is_exact [static], [constexpr], [inherited]
True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

is_iec559

bool std::__numeric_limits_base::is_iec559 [static], [constexpr], [inherited]
True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

is_integer

bool std::__numeric_limits_base::is_integer [static], [constexpr], [inherited]
True if the type is integer.

is_modulo

bool std::__numeric_limits_base::is_modulo [static], [constexpr], [inherited]
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or * on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

is_signed

bool std::__numeric_limits_base::is_signed [static], [constexpr], [inherited]
True if the type is signed.

is_specialized

bool std::__numeric_limits_base::is_specialized [static], [constexpr], [inherited]
This will be true for all fundamental types (which have specializations), and false for everything else.

max_digits10

int std::__numeric_limits_base::max_digits10 [static], [constexpr], [inherited]
The number of base 10 digits required to ensure that values which differ are always differentiated.

max_exponent

int std::__numeric_limits_base::max_exponent [static], [constexpr], [inherited]
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

max_exponent10

int std::__numeric_limits_base::max_exponent10 [static], [constexpr], [inherited]
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

min_exponent

int std::__numeric_limits_base::min_exponent [static], [constexpr], [inherited]
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

min_exponent10

int std::__numeric_limits_base::min_exponent10 [static], [constexpr], [inherited]
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

radix

int std::__numeric_limits_base::radix [static], [constexpr], [inherited]
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

round_style

`float_round_style std::__numeric_limits_base::round_style [static], [constexpr], [inherited]`
See std::float_round_style for more information. This is only meaningful for floating types; integer types will all be round_toward_zero.

tinyness_before

`bool std::__numeric_limits_base::tinyness_before [static], [constexpr], [inherited]`
True if tininess is detected before rounding. (see IEC 559)

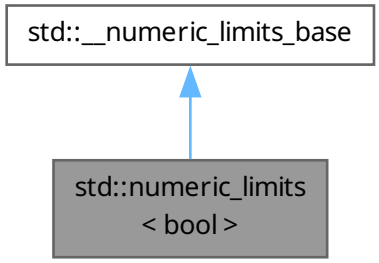
traps

`bool std::__numeric_limits_base::traps [static], [constexpr], [inherited]`
True if trapping is implemented for this type.
The documentation for this struct was generated from the following file:

- limits

5.711 std::numeric_limits< bool > Struct Reference

`#include <limits>`
Inheritance diagram for std::numeric_limits< bool >:



Static Public Member Functions

- static constexpr bool denorm_min () noexcept
- static constexpr bool **denorm_min** () noexcept
- static constexpr bool epsilon () noexcept
- static constexpr bool **epsilon** () noexcept
- static constexpr bool infinity () noexcept
- static constexpr bool **infinity** () noexcept
- static constexpr bool lowest () noexcept
- static constexpr bool **lowest** () noexcept
- static constexpr bool max () noexcept
- static constexpr bool **max** () noexcept
- static constexpr bool min () noexcept
- static constexpr bool **min** () noexcept

- static constexpr bool quiet_NaN () noexcept
- static constexpr bool **quiet_NaN** () noexcept
- static constexpr bool round_error () noexcept
- static constexpr bool **round_error** () noexcept
- static constexpr bool signaling_NaN () noexcept
- static constexpr bool **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int digits
- static constexpr int **digits**
- static constexpr int digits10
- static constexpr int **digits10**
- static constexpr float_denorm_style has_denorm
- static constexpr float_denorm_style **has_denorm**
- static constexpr bool has_denorm_loss
- static constexpr bool **has_denorm_loss**
- static constexpr bool has_infinity
- static constexpr bool **has_infinity**
- static constexpr bool has_quiet_NaN
- static constexpr bool **has_quiet_NaN**
- static constexpr bool has_signaling_NaN
- static constexpr bool **has_signaling_NaN**
- static constexpr bool is_bounded
- static constexpr bool **is_bounded**
- static constexpr bool is_exact
- static constexpr bool **is_exact**
- static constexpr bool is_iec559
- static constexpr bool **is_iec559**
- static constexpr bool is_integer
- static constexpr bool **is_integer**
- static constexpr bool is_modulo
- static constexpr bool **is_modulo**
- static constexpr bool is_signed
- static constexpr bool **is_signed**
- static constexpr bool is_specialized
- static constexpr bool **is_specialized**
- static constexpr int max_digits10
- static constexpr int **max_digits10**
- static constexpr int max_exponent
- static constexpr int **max_exponent**
- static constexpr int max_exponent10
- static constexpr int **max_exponent10**
- static constexpr int min_exponent
- static constexpr int **min_exponent**
- static constexpr int min_exponent10
- static constexpr int **min_exponent10**
- static constexpr int radix
- static constexpr int **radix**
- static constexpr float_round_style round_style
- static constexpr float_round_style **round_style**

- static constexpr bool tinyness_before
- static constexpr bool **tinyness_before**
- static constexpr bool traps
- static constexpr bool **traps**

5.711.1 Detailed Description

numeric_limits<bool> specialization.

5.711.2 Member Function Documentation

denorm_min()

`static constexpr bool std::numeric_limits< bool >::denorm_min () [inline], [static], [constexpr], [noexcept]`
The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

epsilon()

`static constexpr bool std::numeric_limits< bool >::epsilon () [inline], [static], [constexpr], [noexcept]`
The *machine epsilon:* the difference between 1 and the least value greater than 1 that is representable.

infinity()

`static constexpr bool std::numeric_limits< bool >::infinity () [inline], [static], [constexpr], [noexcept]`
The representation of positive infinity, if `has_infinity`.

lowest()

`static constexpr bool std::numeric_limits< bool >::lowest () [inline], [static], [constexpr], [noexcept]`
A finite value x such that there is no other finite value y where y < x.

max()

`static constexpr bool std::numeric_limits< bool >::max () [inline], [static], [constexpr], [noexcept]`
The maximum finite value.

min()

`static constexpr bool std::numeric_limits< bool >::min () [inline], [static], [constexpr], [noexcept]`
The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

quiet_NaN()

`static constexpr bool std::numeric_limits< bool >::quiet_NaN () [inline], [static], [constexpr], [noexcept]`
The representation of a quiet Not a Number, if `has_quiet_NaN`.

round_error()

static constexpr bool std::numeric_limits< bool >::round_error () [inline], [static], [constexpr], [noexcept]
The maximum rounding error measurement (see LIA-1).

signaling_NaN()

static constexpr bool std::numeric_limits< bool >::signaling_NaN () [inline], [static], [constexpr], [noexcept]
The representation of a signaling Not a Number, if has_signaling_NaN.

5.711.3 Member Data Documentation

digits

int std::__numeric_limits_base::digits [static], [constexpr]
The number of radix digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of radix digits in the mantissa.

digits10

int std::__numeric_limits_base::digits10 [static], [constexpr]
The number of base 10 digits that can be represented without change.

has_denorm

float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
See std::float_denorm_style for more information.

has_denorm_loss

bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

has_infinity

bool std::__numeric_limits_base::has_infinity [static], [constexpr]
True if the type has a representation for positive infinity.

has_quiet_NaN

bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
True if the type has a representation for a quiet (non-signaling) Not a Number.

has_signaling_NaN

bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
True if the type has a representation for a signaling Not a Number.

is_bounded

bool std::__numeric_limits_base::is_bounded [static], [constexpr]
True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

is_exact

`bool std::__numeric_limits_base::is_exact [static], [constexpr]`
True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

is_iec559

`bool std::__numeric_limits_base::is_iec559 [static], [constexpr]`
True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

is_integer

`bool std::__numeric_limits_base::is_integer [static], [constexpr]`
True if the type is integer.

is_modulo

`bool std::__numeric_limits_base::is_modulo [static], [constexpr]`
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or $*$ on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

is_signed

`bool std::__numeric_limits_base::is_signed [static], [constexpr]`
True if the type is signed.

is_specialized

`bool std::__numeric_limits_base::is_specialized [static], [constexpr]`
This will be true for all fundamental types (which have specializations), and false for everything else.

max_digits10

`int std::__numeric_limits_base::max_digits10 [static], [constexpr]`
The number of base 10 digits required to ensure that values which differ are always differentiated.

max_exponent

`int std::__numeric_limits_base::max_exponent [static], [constexpr]`
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

max_exponent10

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr]`
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

min_exponent

```
int std::__numeric_limits_base::min_exponent  [static], [constexpr]
```
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

min_exponent10

```
int std::__numeric_limits_base::min_exponent10  [static], [constexpr]
```
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

radix

```
int std::__numeric_limits_base::radix  [static], [constexpr]
```
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

round_style

```
float_round_style std::__numeric_limits_base::round_style  [static], [constexpr]
```
See std::float_round_style for more information. This is only meaningful for floating types; integer types will all be round_toward_zero.

tinyness_before

```
bool std::__numeric_limits_base::tinyness_before  [static], [constexpr]
```
True if tininess is detected before rounding. (see IEC 559)

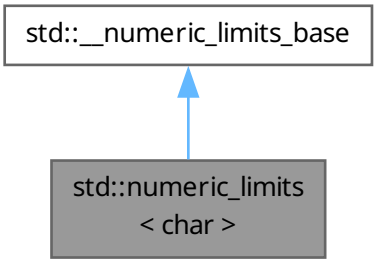
traps

```
bool std::__numeric_limits_base::traps  [static], [constexpr]
```
True if trapping is implemented for this type.
The documentation for this struct was generated from the following file:

- limits

5.712 std::numeric_limits< char > Struct Reference

`#include <limits>`
Inheritance diagram for std::numeric_limits< char >:



Static Public Member Functions

- static constexpr char denorm_min () noexcept
- static constexpr char **denorm_min** () noexcept
- static constexpr char epsilon () noexcept
- static constexpr char **epsilon** () noexcept
- static constexpr char infinity () noexcept
- static constexpr char **infinity** () noexcept
- static constexpr char lowest () noexcept
- static constexpr char **lowest** () noexcept
- static constexpr char max () noexcept
- static constexpr char **max** () noexcept
- static constexpr char min () noexcept
- static constexpr char **min** () noexcept
- static constexpr char quiet_NaN () noexcept
- static constexpr char **quiet_NaN** () noexcept
- static constexpr char round_error () noexcept
- static constexpr char **round_error** () noexcept
- static constexpr char signaling_NaN () noexcept
- static constexpr char **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int digits
- static constexpr int **digits**
- static constexpr int digits10
- static constexpr int **digits10**
- static constexpr float_denorm_style has_denorm
- static constexpr float_denorm_style **has_denorm**
- static constexpr bool has_denorm_loss
- static constexpr bool **has_denorm_loss**
- static constexpr bool has_infinity
- static constexpr bool **has_infinity**
- static constexpr bool has_quiet_NaN
- static constexpr bool **has_quiet_NaN**
- static constexpr bool has_signaling_NaN
- static constexpr bool **has_signaling_NaN**
- static constexpr bool is_bounded
- static constexpr bool **is_bounded**
- static constexpr bool is_exact
- static constexpr bool **is_exact**
- static constexpr bool is_iec559
- static constexpr bool **is_iec559**
- static constexpr bool is_integer
- static constexpr bool **is_integer**
- static constexpr bool is_modulo
- static constexpr bool **is_modulo**
- static constexpr bool is_signed
- static constexpr bool **is_signed**
- static constexpr bool is_specialized
- static constexpr bool **is_specialized**
- static constexpr int max_digits10

- static constexpr int **max_digits10**
- static constexpr int max_exponent
- static constexpr int **max_exponent**
- static constexpr int max_exponent10
- static constexpr int **max_exponent10**
- static constexpr int min_exponent
- static constexpr int **min_exponent**
- static constexpr int min_exponent10
- static constexpr int **min_exponent10**
- static constexpr int radix
- static constexpr int **radix**
- static constexpr float_round_style round_style
- static constexpr float_round_style **round_style**
- static constexpr bool tinyness_before
- static constexpr bool **tinyness_before**
- static constexpr bool traps
- static constexpr bool **traps**

5.712.1 Detailed Description

numeric_limits<char> specialization.

5.712.2 Member Function Documentation

denorm_min()

`static constexpr char std::numeric_limits< char >::denorm_min () [inline], [static], [constexpr], [noexcept]`

The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

epsilon()

`static constexpr char std::numeric_limits< char >::epsilon () [inline], [static], [constexpr], [noexcept]`

The *machine epsilon:* the difference between 1 and the least value greater than 1 that is representable.

infinity()

`static constexpr char std::numeric_limits< char >::infinity () [inline], [static], [constexpr], [noexcept]`

The representation of positive infinity, if `has_infinity`.

lowest()

`static constexpr char std::numeric_limits< char >::lowest () [inline], [static], [constexpr], [noexcept]`

A finite value x such that there is no other finite value y where y < x.

max()

`static constexpr char std::numeric_limits< char >::max () [inline], [static], [constexpr], [noexcept]`

The maximum finite value.

min()

`static constexpr char std::numeric_limits< char >::min () [inline], [static], [constexpr], [noexcept]`
The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

quiet_NaN()

`static constexpr char std::numeric_limits< char >::quiet_NaN () [inline], [static], [constexpr], [noexcept]`
The representation of a quiet Not a Number, if `has_quiet_NaN`.

round_error()

`static constexpr char std::numeric_limits< char >::round_error () [inline], [static], [constexpr], [noexcept]`
The maximum rounding error measurement (see LIA-1).

signaling_NaN()

`static constexpr char std::numeric_limits< char >::signaling_NaN () [inline], [static], [constexpr], [noexcept]`
The representation of a signaling Not a Number, if `has_signaling_NaN`.

5.712.3 Member Data Documentation

digits

`int std::__numeric_limits_base::digits [static], [constexpr]`
The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

digits10

`int std::__numeric_limits_base::digits10 [static], [constexpr]`
The number of base 10 digits that can be represented without change.

has_denorm

`float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]`
See std::float_denorm_style for more information.

has_denorm_loss

`bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]`
True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

has_infinity

`bool std::__numeric_limits_base::has_infinity [static], [constexpr]`
True if the type has a representation for positive infinity.

has_quiet_NaN

`bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]`
True if the type has a representation for a quiet (non-signaling) Not a Number.

has_signaling_NaN

bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
True if the type has a representation for a signaling Not a Number.

is_bounded

bool std::__numeric_limits_base::is_bounded [static], [constexpr]
True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

is_exact

bool std::__numeric_limits_base::is_exact [static], [constexpr]
True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

is_iec559

bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

is_integer

bool std::__numeric_limits_base::is_integer [static], [constexpr]
True if the type is integer.

is_modulo

bool std::__numeric_limits_base::is_modulo [static], [constexpr]
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or $*$ on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

is_signed

bool std::__numeric_limits_base::is_signed [static], [constexpr]
True if the type is signed.

is_specialized

bool std::__numeric_limits_base::is_specialized [static], [constexpr]
This will be true for all fundamental types (which have specializations), and false for everything else.

max_digits10

int std::__numeric_limits_base::max_digits10 [static], [constexpr]
The number of base 10 digits required to ensure that values which differ are always differentiated.

max_exponent

int std::__numeric_limits_base::max_exponent [static], [constexpr]
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

max_exponent10

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr]`
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

min_exponent

`int std::__numeric_limits_base::min_exponent [static], [constexpr]`
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

min_exponent10

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr]`
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

radix

`int std::__numeric_limits_base::radix [static], [constexpr]`
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

round_style

`float_round_style std::__numeric_limits_base::round_style [static], [constexpr]`
See std::float_round_style for more information. This is only meaningful for floating types; integer types will all be round_toward_zero.

tinyness_before

`bool std::__numeric_limits_base::tinyness_before [static], [constexpr]`
True if tininess is detected before rounding. (see IEC 559)

traps

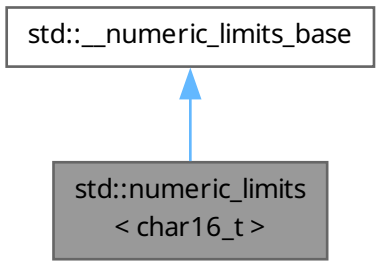
`bool std::__numeric_limits_base::traps [static], [constexpr]`
True if trapping is implemented for this type.
The documentation for this struct was generated from the following file:

- limits

5.713 std::numeric_limits< char16_t > Struct Reference

`#include <limits>`

Inheritance diagram for std::numeric_limits< char16_t >:



Static Public Member Functions

- static constexpr char16_t denorm_min () noexcept
- static constexpr char16_t **denorm_min** () noexcept
- static constexpr char16_t epsilon () noexcept
- static constexpr char16_t **epsilon** () noexcept
- static constexpr char16_t infinity () noexcept
- static constexpr char16_t **infinity** () noexcept
- static constexpr char16_t lowest () noexcept
- static constexpr char16_t **lowest** () noexcept
- static constexpr char16_t max () noexcept
- static constexpr char16_t **max** () noexcept
- static constexpr char16_t min () noexcept
- static constexpr char16_t **min** () noexcept
- static constexpr char16_t quiet_NaN () noexcept
- static constexpr char16_t **quiet_NaN** () noexcept
- static constexpr char16_t round_error () noexcept
- static constexpr char16_t **round_error** () noexcept
- static constexpr char16_t signaling_NaN () noexcept
- static constexpr char16_t **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int digits
- static constexpr int **digits**
- static constexpr int digits10
- static constexpr int **digits10**
- static constexpr float_denorm_style has_denorm
- static constexpr float_denorm_style **has_denorm**
- static constexpr bool has_denorm_loss
- static constexpr bool **has_denorm_loss**
- static constexpr bool has_infinity
- static constexpr bool **has_infinity**
- static constexpr bool has_quiet_NaN

- static constexpr bool **has_quiet_NaN**
- static constexpr bool has_signaling_NaN
- static constexpr bool **has_signaling_NaN**
- static constexpr bool is_bounded
- static constexpr bool **is_bounded**
- static constexpr bool is_exact
- static constexpr bool **is_exact**
- static constexpr bool is_iec559
- static constexpr bool **is_iec559**
- static constexpr bool is_integer
- static constexpr bool **is_integer**
- static constexpr bool is_modulo
- static constexpr bool **is_modulo**
- static constexpr bool is_signed
- static constexpr bool **is_signed**
- static constexpr bool is_specialized
- static constexpr bool **is_specialized**
- static constexpr int max_digits10
- static constexpr int **max_digits10**
- static constexpr int max_exponent
- static constexpr int **max_exponent**
- static constexpr int max_exponent10
- static constexpr int **max_exponent10**
- static constexpr int min_exponent
- static constexpr int **min_exponent**
- static constexpr int min_exponent10
- static constexpr int **min_exponent10**
- static constexpr int radix
- static constexpr int **radix**
- static constexpr float_round_style round_style
- static constexpr float_round_style **round_style**
- static constexpr bool tinyness_before
- static constexpr bool **tinyness_before**
- static constexpr bool traps
- static constexpr bool **traps**

5.713.1 Detailed Description

numeric_limits<char16_t> specialization.

5.713.2 Member Function Documentation

denorm_min()

```
static constexpr char16_t std::numeric_limits< char16_t >::denorm_min ()  [inline], [static],
[constexpr], [noexcept]
```

The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

epsilon()

```
static constexpr char16_t std::numeric_limits< char16_t >::epsilon ()  [inline], [static], [constexpr],
[noexcept]
```

The *machine epsilon:* the difference between 1 and the least value greater than 1 that is representable.

infinity()

```
static constexpr char16_t std::numeric_limits< char16_t >::infinity ()  [inline], [static], [constexpr], [noexcept]
```
The representation of positive infinity, if `has_infinity`.

lowest()

```
static constexpr char16_t std::numeric_limits< char16_t >::lowest ()  [inline], [static], [constexpr], [noexcept]
```
A finite value x such that there is no other finite value y where y < x.

max()

```
static constexpr char16_t std::numeric_limits< char16_t >::max ()  [inline], [static], [constexpr], [noexcept]
```
The maximum finite value.

min()

```
static constexpr char16_t std::numeric_limits< char16_t >::min ()  [inline], [static], [constexpr], [noexcept]
```
The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

quiet_NaN()

```
static constexpr char16_t std::numeric_limits< char16_t >::quiet_NaN ()  [inline], [static], [constexpr], [noexcept]
```
The representation of a quiet Not a Number, if `has_quiet_NaN`.

round_error()

```
static constexpr char16_t std::numeric_limits< char16_t >::round_error ()  [inline], [static], [constexpr], [noexcept]
```
The maximum rounding error measurement (see LIA-1).

signaling_NaN()

```
static constexpr char16_t std::numeric_limits< char16_t >::signaling_NaN ()  [inline], [static], [constexpr], [noexcept]
```
The representation of a signaling Not a Number, if `has_signaling_NaN`.

5.713.3 Member Data Documentation

digits

```
int std::__numeric_limits_base::digits  [static], [constexpr]
```
The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

digits10

```
int std::__numeric_limits_base::digits10  [static], [constexpr]
```
The number of base 10 digits that can be represented without change.

has_denorm

`float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]`
See std::float_denorm_style for more information.

has_denorm_loss

`bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]`
True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

has_infinity

`bool std::__numeric_limits_base::has_infinity [static], [constexpr]`
True if the type has a representation for positive infinity.

has_quiet_NaN

`bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]`
True if the type has a representation for a quiet (non-signaling) Not a Number.

has_signaling_NaN

`bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]`
True if the type has a representation for a signaling Not a Number.

is_bounded

`bool std::__numeric_limits_base::is_bounded [static], [constexpr]`
True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

is_exact

`bool std::__numeric_limits_base::is_exact [static], [constexpr]`
True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

is_iec559

`bool std::__numeric_limits_base::is_iec559 [static], [constexpr]`
True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

is_integer

`bool std::__numeric_limits_base::is_integer [static], [constexpr]`
True if the type is integer.

is_modulo

`bool std::__numeric_limits_base::is_modulo [static], [constexpr]`
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or $*$ on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

is_signed

bool std::__numeric_limits_base::is_signed [static], [constexpr]
True if the type is signed.

is_specialized

bool std::__numeric_limits_base::is_specialized [static], [constexpr]
This will be true for all fundamental types (which have specializations), and false for everything else.

max_digits10

int std::__numeric_limits_base::max_digits10 [static], [constexpr]
The number of base 10 digits required to ensure that values which differ are always differentiated.

max_exponent

int std::__numeric_limits_base::max_exponent [static], [constexpr]
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

max_exponent10

int std::__numeric_limits_base::max_exponent10 [static], [constexpr]
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

min_exponent

int std::__numeric_limits_base::min_exponent [static], [constexpr]
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

min_exponent10

int std::__numeric_limits_base::min_exponent10 [static], [constexpr]
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

radix

int std::__numeric_limits_base::radix [static], [constexpr]
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

round_style

float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
See std::float_round_style for more information. This is only meaningful for floating types; integer types will all be round_toward_zero.

tinyness_before

bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
True if tininess is detected before rounding. (see IEC 559)

traps

```
bool std::__numeric_limits_base::traps  [static], [constexpr]
```

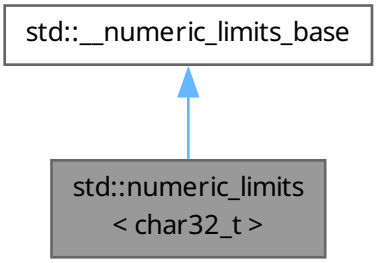
True if trapping is implemented for this type.
The documentation for this struct was generated from the following file:

- limits

5.714 std::numeric_limits< char32_t > Struct Reference

```
#include <limits>
```

Inheritance diagram for std::numeric_limits< char32_t >:



Static Public Member Functions

- static constexpr char32_t denorm_min () noexcept
- static constexpr char32_t **denorm_min** () noexcept
- static constexpr char32_t epsilon () noexcept
- static constexpr char32_t **epsilon** () noexcept
- static constexpr char32_t infinity () noexcept
- static constexpr char32_t **infinity** () noexcept
- static constexpr char32_t lowest () noexcept
- static constexpr char32_t **lowest** () noexcept
- static constexpr char32_t max () noexcept
- static constexpr char32_t **max** () noexcept
- static constexpr char32_t min () noexcept
- static constexpr char32_t **min** () noexcept
- static constexpr char32_t quiet_NaN () noexcept
- static constexpr char32_t **quiet_NaN** () noexcept
- static constexpr char32_t round_error () noexcept
- static constexpr char32_t **round_error** () noexcept
- static constexpr char32_t signaling_NaN () noexcept
- static constexpr char32_t **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int digits
- static constexpr int **digits**
- static constexpr int digits10
- static constexpr int **digits10**
- static constexpr float_denorm_style has_denorm
- static constexpr float_denorm_style **has_denorm**
- static constexpr bool has_denorm_loss
- static constexpr bool **has_denorm_loss**
- static constexpr bool has_infinity
- static constexpr bool **has_infinity**
- static constexpr bool has_quiet_NaN
- static constexpr bool **has_quiet_NaN**
- static constexpr bool has_signaling_NaN
- static constexpr bool **has_signaling_NaN**
- static constexpr bool is_bounded
- static constexpr bool **is_bounded**
- static constexpr bool is_exact
- static constexpr bool **is_exact**
- static constexpr bool is_iec559
- static constexpr bool **is_iec559**
- static constexpr bool is_integer
- static constexpr bool **is_integer**
- static constexpr bool is_modulo
- static constexpr bool **is_modulo**
- static constexpr bool is_signed
- static constexpr bool **is_signed**
- static constexpr bool is_specialized
- static constexpr bool **is_specialized**
- static constexpr int max_digits10
- static constexpr int **max_digits10**
- static constexpr int max_exponent
- static constexpr int **max_exponent**
- static constexpr int max_exponent10
- static constexpr int **max_exponent10**
- static constexpr int min_exponent
- static constexpr int **min_exponent**
- static constexpr int min_exponent10
- static constexpr int **min_exponent10**
- static constexpr int radix
- static constexpr int **radix**
- static constexpr float_round_style round_style
- static constexpr float_round_style **round_style**
- static constexpr bool tinyness_before
- static constexpr bool **tinyness_before**
- static constexpr bool traps
- static constexpr bool **traps**

5.714.1 Detailed Description

numeric_limits<char32_t> specialization.

5.714.2 Member Function Documentation

denorm_min()

`static constexpr char32_t std::numeric_limits< char32_t >::denorm_min () [inline], [static], [constexpr], [noexcept]`
The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

epsilon()

`static constexpr char32_t std::numeric_limits< char32_t >::epsilon () [inline], [static], [constexpr], [noexcept]`
The *machine epsilon:* the difference between 1 and the least value greater than 1 that is representable.

infinity()

`static constexpr char32_t std::numeric_limits< char32_t >::infinity () [inline], [static], [constexpr], [noexcept]`
The representation of positive infinity, if `has_infinity`.

lowest()

`static constexpr char32_t std::numeric_limits< char32_t >::lowest () [inline], [static], [constexpr], [noexcept]`
A finite value x such that there is no other finite value y where y $<$ x.

max()

`static constexpr char32_t std::numeric_limits< char32_t >::max () [inline], [static], [constexpr], [noexcept]`
The maximum finite value.

min()

`static constexpr char32_t std::numeric_limits< char32_t >::min () [inline], [static], [constexpr], [noexcept]`
The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

quiet_NaN()

`static constexpr char32_t std::numeric_limits< char32_t >::quiet_NaN () [inline], [static], [constexpr], [noexcept]`
The representation of a quiet Not a Number, if `has_quiet_NaN`.

round_error()

`static constexpr char32_t std::numeric_limits< char32_t >::round_error () [inline], [static], [constexpr], [noexcept]`
The maximum rounding error measurement (see LIA-1).

signaling_NaN()

`static constexpr char32_t std::numeric_limits< char32_t >::signaling_NaN () [inline], [static], [constexpr], [noexcept]`
The representation of a signaling Not a Number, if `has_signaling_NaN`.

5.714.3 Member Data Documentation

digits

```
int std::__numeric_limits_base::digits  [static], [constexpr]
```
The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

digits10

```
int std::__numeric_limits_base::digits10  [static], [constexpr]
```
The number of base 10 digits that can be represented without change.

has_denorm

```
float_denorm_style std::__numeric_limits_base::has_denorm  [static], [constexpr]
```
See std::float_denorm_style for more information.

has_denorm_loss

```
bool std::__numeric_limits_base::has_denorm_loss  [static], [constexpr]
```
True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

has_infinity

```
bool std::__numeric_limits_base::has_infinity  [static], [constexpr]
```
True if the type has a representation for positive infinity.

has_quiet_NaN

```
bool std::__numeric_limits_base::has_quiet_NaN  [static], [constexpr]
```
True if the type has a representation for a quiet (non-signaling) Not a Number.

has_signaling_NaN

```
bool std::__numeric_limits_base::has_signaling_NaN  [static], [constexpr]
```
True if the type has a representation for a signaling Not a Number.

is_bounded

```
bool std::__numeric_limits_base::is_bounded  [static], [constexpr]
```
True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

is_exact

```
bool std::__numeric_limits_base::is_exact  [static], [constexpr]
```
True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

is_iec559

```
bool std::__numeric_limits_base::is_iec559  [static], [constexpr]
```
True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

is_integer

`bool std::__numeric_limits_base::is_integer [static], [constexpr]`
True if the type is integer.

is_modulo

`bool std::__numeric_limits_base::is_modulo [static], [constexpr]`
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or $*$ on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

is_signed

`bool std::__numeric_limits_base::is_signed [static], [constexpr]`
True if the type is signed.

is_specialized

`bool std::__numeric_limits_base::is_specialized [static], [constexpr]`
This will be true for all fundamental types (which have specializations), and false for everything else.

max_digits10

`int std::__numeric_limits_base::max_digits10 [static], [constexpr]`
The number of base 10 digits required to ensure that values which differ are always differentiated.

max_exponent

`int std::__numeric_limits_base::max_exponent [static], [constexpr]`
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

max_exponent10

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr]`
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

min_exponent

`int std::__numeric_limits_base::min_exponent [static], [constexpr]`
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

min_exponent10

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr]`
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

radix

`int std::__numeric_limits_base::radix [static], [constexpr]`
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

round_style

```
float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
```
See std::float_round_style for more information. This is only meaningful for floating types; integer types will all be round_toward_zero.

tinyness_before

```
bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
```
True if tininess is detected before rounding. (see IEC 559)

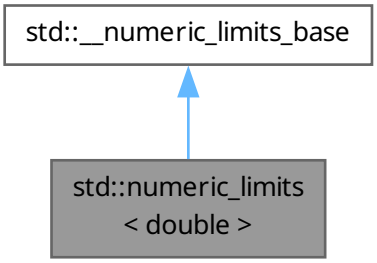
traps

```
bool std::__numeric_limits_base::traps [static], [constexpr]
```
True if trapping is implemented for this type.
The documentation for this struct was generated from the following file:

- limits

5.715 std::numeric_limits< double > Struct Reference

```
#include <limits>
```
Inheritance diagram for std::numeric_limits< double >:



Static Public Member Functions

- static constexpr double denorm_min () noexcept
- static constexpr double **denorm_min** () noexcept
- static constexpr double epsilon () noexcept
- static constexpr double **epsilon** () noexcept
- static constexpr double infinity () noexcept
- static constexpr double **infinity** () noexcept
- static constexpr double lowest () noexcept
- static constexpr double **lowest** () noexcept
- static constexpr double max () noexcept
- static constexpr double **max** () noexcept
- static constexpr double min () noexcept
- static constexpr double **min** () noexcept

- static constexpr double quiet_NaN () noexcept
- static constexpr double **quiet_NaN** () noexcept
- static constexpr double round_error () noexcept
- static constexpr double **round_error** () noexcept
- static constexpr double signaling_NaN () noexcept
- static constexpr double **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int digits
- static constexpr int **digits**
- static constexpr int digits10
- static constexpr int **digits10**
- static constexpr float_denorm_style has_denorm
- static constexpr float_denorm_style **has_denorm**
- static constexpr bool has_denorm_loss
- static constexpr bool **has_denorm_loss**
- static constexpr bool has_infinity
- static constexpr bool **has_infinity**
- static constexpr bool has_quiet_NaN
- static constexpr bool **has_quiet_NaN**
- static constexpr bool has_signaling_NaN
- static constexpr bool **has_signaling_NaN**
- static constexpr bool is_bounded
- static constexpr bool **is_bounded**
- static constexpr bool is_exact
- static constexpr bool **is_exact**
- static constexpr bool is_iec559
- static constexpr bool **is_iec559**
- static constexpr bool is_integer
- static constexpr bool **is_integer**
- static constexpr bool is_modulo
- static constexpr bool **is_modulo**
- static constexpr bool is_signed
- static constexpr bool **is_signed**
- static constexpr bool is_specialized
- static constexpr bool **is_specialized**
- static constexpr int max_digits10
- static constexpr int **max_digits10**
- static constexpr int max_exponent
- static constexpr int **max_exponent**
- static constexpr int max_exponent10
- static constexpr int **max_exponent10**
- static constexpr int min_exponent
- static constexpr int **min_exponent**
- static constexpr int min_exponent10
- static constexpr int **min_exponent10**
- static constexpr int radix
- static constexpr int **radix**
- static constexpr float_round_style round_style
- static constexpr float_round_style **round_style**

- static constexpr bool tinyness_before
- static constexpr bool **tinyness_before**
- static constexpr bool traps
- static constexpr bool **traps**

5.715.1 Detailed Description

numeric_limits<double> specialization.

5.715.2 Member Function Documentation

denorm_min()

```
static constexpr double std::numeric_limits< double >::denorm_min ()  [inline], [static], [constexpr], [noexcept]
```

The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

epsilon()

```
static constexpr double std::numeric_limits< double >::epsilon ()  [inline], [static], [constexpr], [noexcept]
```

The *machine epsilon:* the difference between 1 and the least value greater than 1 that is representable.

infinity()

```
static constexpr double std::numeric_limits< double >::infinity ()  [inline], [static], [constexpr], [noexcept]
```

The representation of positive infinity, if `has_infinity`.

lowest()

```
static constexpr double std::numeric_limits< double >::lowest ()  [inline], [static], [constexpr], [noexcept]
```

A finite value x such that there is no other finite value y where y < x.

max()

```
static constexpr double std::numeric_limits< double >::max ()  [inline], [static], [constexpr], [noexcept]
```

The maximum finite value.

min()

```
static constexpr double std::numeric_limits< double >::min ()  [inline], [static], [constexpr], [noexcept]
```

The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

quiet_NaN()

```
static constexpr double std::numeric_limits< double >::quiet_NaN ()  [inline], [static], [constexpr], [noexcept]
```

The representation of a quiet Not a Number, if `has_quiet_NaN`.

round_error()

```
static constexpr double std::numeric_limits< double >::round_error ()  [inline], [static], [constexpr], [noexcept]
```
The maximum rounding error measurement (see LIA-1).

signaling_NaN()

```
static constexpr double std::numeric_limits< double >::signaling_NaN ()  [inline], [static], [constexpr], [noexcept]
```
The representation of a signaling Not a Number, if `has_signaling_NaN`.

5.715.3 Member Data Documentation

digits

```
int std::__numeric_limits_base::digits  [static], [constexpr]
```
The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

digits10

```
int std::__numeric_limits_base::digits10  [static], [constexpr]
```
The number of base 10 digits that can be represented without change.

has_denorm

```
float_denorm_style std::__numeric_limits_base::has_denorm  [static], [constexpr]
```
See std::float_denorm_style for more information.

has_denorm_loss

```
bool std::__numeric_limits_base::has_denorm_loss  [static], [constexpr]
```
True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

has_infinity

```
bool std::__numeric_limits_base::has_infinity  [static], [constexpr]
```
True if the type has a representation for positive infinity.

has_quiet_NaN

```
bool std::__numeric_limits_base::has_quiet_NaN  [static], [constexpr]
```
True if the type has a representation for a quiet (non-signaling) Not a Number.

has_signaling_NaN

```
bool std::__numeric_limits_base::has_signaling_NaN  [static], [constexpr]
```
True if the type has a representation for a signaling Not a Number.

is_bounded

```
bool std::__numeric_limits_base::is_bounded  [static], [constexpr]
```
True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

is_exact

bool std::__numeric_limits_base::is_exact [static], [constexpr]
True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

is_iec559

bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

is_integer

bool std::__numeric_limits_base::is_integer [static], [constexpr]
True if the type is integer.

is_modulo

bool std::__numeric_limits_base::is_modulo [static], [constexpr]
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or $*$ on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

is_signed

bool std::__numeric_limits_base::is_signed [static], [constexpr]
True if the type is signed.

is_specialized

bool std::__numeric_limits_base::is_specialized [static], [constexpr]
This will be true for all fundamental types (which have specializations), and false for everything else.

max_digits10

int std::__numeric_limits_base::max_digits10 [static], [constexpr]
The number of base 10 digits required to ensure that values which differ are always differentiated.

max_exponent

int std::__numeric_limits_base::max_exponent [static], [constexpr]
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

max_exponent10

int std::__numeric_limits_base::max_exponent10 [static], [constexpr]
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

min_exponent

`int std::__numeric_limits_base::min_exponent [static], [constexpr]`
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

min_exponent10

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr]`
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

radix

`int std::__numeric_limits_base::radix [static], [constexpr]`
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

round_style

`float_round_style std::__numeric_limits_base::round_style [static], [constexpr]`
See std::float_round_style for more information. This is only meaningful for floating types; integer types will all be round_toward_zero.

tinyness_before

`bool std::__numeric_limits_base::tinyness_before [static], [constexpr]`
True if tininess is detected before rounding. (see IEC 559)

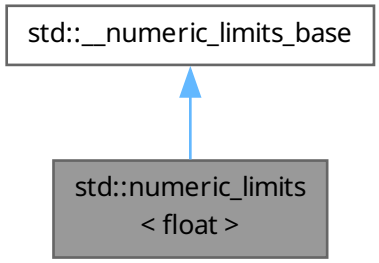
traps

`bool std::__numeric_limits_base::traps [static], [constexpr]`
True if trapping is implemented for this type.
The documentation for this struct was generated from the following file:

- limits

5.716 std::numeric_limits< float > Struct Reference

`#include <limits>`
Inheritance diagram for std::numeric_limits< float >:



Static Public Member Functions

- static constexpr float denorm_min () noexcept
- static constexpr float **denorm_min** () noexcept
- static constexpr float epsilon () noexcept
- static constexpr float **epsilon** () noexcept
- static constexpr float infinity () noexcept
- static constexpr float **infinity** () noexcept
- static constexpr float lowest () noexcept
- static constexpr float **lowest** () noexcept
- static constexpr float max () noexcept
- static constexpr float **max** () noexcept
- static constexpr float min () noexcept
- static constexpr float **min** () noexcept
- static constexpr float quiet_NaN () noexcept
- static constexpr float **quiet_NaN** () noexcept
- static constexpr float round_error () noexcept
- static constexpr float **round_error** () noexcept
- static constexpr float signaling_NaN () noexcept
- static constexpr float **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int digits
- static constexpr int **digits**
- static constexpr int digits10
- static constexpr int **digits10**
- static constexpr float_denorm_style has_denorm
- static constexpr float_denorm_style **has_denorm**
- static constexpr bool has_denorm_loss
- static constexpr bool **has_denorm_loss**
- static constexpr bool has_infinity
- static constexpr bool **has_infinity**
- static constexpr bool has_quiet_NaN
- static constexpr bool **has_quiet_NaN**
- static constexpr bool has_signaling_NaN
- static constexpr bool **has_signaling_NaN**
- static constexpr bool is_bounded
- static constexpr bool **is_bounded**
- static constexpr bool is_exact
- static constexpr bool **is_exact**
- static constexpr bool is_iec559
- static constexpr bool **is_iec559**
- static constexpr bool is_integer
- static constexpr bool **is_integer**
- static constexpr bool is_modulo
- static constexpr bool **is_modulo**
- static constexpr bool is_signed
- static constexpr bool **is_signed**
- static constexpr bool is_specialized
- static constexpr bool **is_specialized**
- static constexpr int max_digits10

- static constexpr int **max_digits10**
- static constexpr int max_exponent
- static constexpr int **max_exponent**
- static constexpr int max_exponent10
- static constexpr int **max_exponent10**
- static constexpr int min_exponent
- static constexpr int **min_exponent**
- static constexpr int min_exponent10
- static constexpr int **min_exponent10**
- static constexpr int radix
- static constexpr int **radix**
- static constexpr float_round_style round_style
- static constexpr float_round_style **round_style**
- static constexpr bool tinyness_before
- static constexpr bool **tinyness_before**
- static constexpr bool traps
- static constexpr bool **traps**

5.716.1 Detailed Description

numeric_limits<float> specialization.

5.716.2 Member Function Documentation

denorm_min()

`static constexpr float std::numeric_limits< float >::denorm_min () [inline], [static], [constexpr], [noexcept]`

The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

epsilon()

`static constexpr float std::numeric_limits< float >::epsilon () [inline], [static], [constexpr], [noexcept]`

The *machine epsilon:* the difference between 1 and the least value greater than 1 that is representable.

infinity()

`static constexpr float std::numeric_limits< float >::infinity () [inline], [static], [constexpr], [noexcept]`

The representation of positive infinity, if `has_infinity`.

lowest()

`static constexpr float std::numeric_limits< float >::lowest () [inline], [static], [constexpr], [noexcept]`

A finite value x such that there is no other finite value y where y < x.

max()

`static constexpr float std::numeric_limits< float >::max () [inline], [static], [constexpr], [noexcept]`

The maximum finite value.

min()

static constexpr float std::numeric_limits< float >::min () [inline], [static], [constexpr], [noexcept]
The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

quiet_NaN()

static constexpr float std::numeric_limits< float >::quiet_NaN () [inline], [static], [constexpr], [noexcept]
The representation of a quiet Not a Number, if has_quiet_NaN.

round_error()

static constexpr float std::numeric_limits< float >::round_error () [inline], [static], [constexpr], [noexcept]
The maximum rounding error measurement (see LIA-1).

signaling_NaN()

static constexpr float std::numeric_limits< float >::signaling_NaN () [inline], [static], [constexpr], [noexcept]
The representation of a signaling Not a Number, if has_signaling_NaN.

5.716.3 Member Data Documentation

digits

int std::__numeric_limits_base::digits [static], [constexpr]
The number of radix digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of radix digits in the mantissa.

digits10

int std::__numeric_limits_base::digits10 [static], [constexpr]
The number of base 10 digits that can be represented without change.

has_denorm

float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
See std::float_denorm_style for more information.

has_denorm_loss

bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

has_infinity

bool std::__numeric_limits_base::has_infinity [static], [constexpr]
True if the type has a representation for positive infinity.

has_quiet_NaN

bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
True if the type has a representation for a quiet (non-signaling) Not a Number.

has_signaling_NaN

`bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]`
True if the type has a representation for a signaling Not a Number.

is_bounded

`bool std::__numeric_limits_base::is_bounded [static], [constexpr]`
True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

is_exact

`bool std::__numeric_limits_base::is_exact [static], [constexpr]`
True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

is_iec559

`bool std::__numeric_limits_base::is_iec559 [static], [constexpr]`
True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

is_integer

`bool std::__numeric_limits_base::is_integer [static], [constexpr]`
True if the type is integer.

is_modulo

`bool std::__numeric_limits_base::is_modulo [static], [constexpr]`
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or $*$ on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

is_signed

`bool std::__numeric_limits_base::is_signed [static], [constexpr]`
True if the type is signed.

is_specialized

`bool std::__numeric_limits_base::is_specialized [static], [constexpr]`
This will be true for all fundamental types (which have specializations), and false for everything else.

max_digits10

`int std::__numeric_limits_base::max_digits10 [static], [constexpr]`
The number of base 10 digits required to ensure that values which differ are always differentiated.

max_exponent

`int std::__numeric_limits_base::max_exponent [static], [constexpr]`
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

max_exponent10

int std::__numeric_limits_base::max_exponent10 [static], [constexpr]
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

min_exponent

int std::__numeric_limits_base::min_exponent [static], [constexpr]
The minimum negative integer such that radix raised to the power of (one less than that integer) is a normalized floating point number.

min_exponent10

int std::__numeric_limits_base::min_exponent10 [static], [constexpr]
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

radix

int std::__numeric_limits_base::radix [static], [constexpr]
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

round_style

float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
See std::float_round_style for more information. This is only meaningful for floating types; integer types will all be round_toward_zero.

tinyness_before

bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
True if tininess is detected before rounding. (see IEC 559)

traps

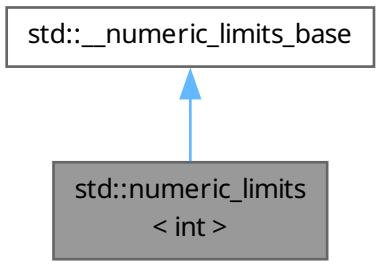
bool std::__numeric_limits_base::traps [static], [constexpr]
True if trapping is implemented for this type.
The documentation for this struct was generated from the following file:

- limits

5.717 std::numeric_limits< int > Struct Reference

#include <limits>

Inheritance diagram for std::numeric_limits< int >:



Static Public Member Functions

- static constexpr int denorm_min () noexcept
- static constexpr int **denorm_min** () noexcept
- static constexpr int epsilon () noexcept
- static constexpr int **epsilon** () noexcept
- static constexpr int infinity () noexcept
- static constexpr int **infinity** () noexcept
- static constexpr int lowest () noexcept
- static constexpr int **lowest** () noexcept
- static constexpr int max () noexcept
- static constexpr int **max** () noexcept
- static constexpr int min () noexcept
- static constexpr int **min** () noexcept
- static constexpr int quiet_NaN () noexcept
- static constexpr int **quiet_NaN** () noexcept
- static constexpr int round_error () noexcept
- static constexpr int **round_error** () noexcept
- static constexpr int signaling_NaN () noexcept
- static constexpr int **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int digits
- static constexpr int **digits**
- static constexpr int digits10
- static constexpr int **digits10**
- static constexpr float_denorm_style has_denorm
- static constexpr float_denorm_style **has_denorm**
- static constexpr bool has_denorm_loss
- static constexpr bool **has_denorm_loss**
- static constexpr bool has_infinity
- static constexpr bool **has_infinity**
- static constexpr bool has_quiet_NaN

- static constexpr bool **has_quiet_NaN**
- static constexpr bool has_signaling_NaN
- static constexpr bool **has_signaling_NaN**
- static constexpr bool is_bounded
- static constexpr bool **is_bounded**
- static constexpr bool is_exact
- static constexpr bool **is_exact**
- static constexpr bool is_iec559
- static constexpr bool **is_iec559**
- static constexpr bool is_integer
- static constexpr bool **is_integer**
- static constexpr bool is_modulo
- static constexpr bool **is_modulo**
- static constexpr bool is_signed
- static constexpr bool **is_signed**
- static constexpr bool is_specialized
- static constexpr bool **is_specialized**
- static constexpr int max_digits10
- static constexpr int **max_digits10**
- static constexpr int max_exponent
- static constexpr int **max_exponent**
- static constexpr int max_exponent10
- static constexpr int **max_exponent10**
- static constexpr int min_exponent
- static constexpr int **min_exponent**
- static constexpr int min_exponent10
- static constexpr int **min_exponent10**
- static constexpr int radix
- static constexpr int **radix**
- static constexpr float_round_style round_style
- static constexpr float_round_style **round_style**
- static constexpr bool tinyness_before
- static constexpr bool **tinyness_before**
- static constexpr bool traps
- static constexpr bool **traps**

5.717.1 Detailed Description

numeric_limits<int> specialization.

5.717.2 Member Function Documentation

denorm_min()

```
static constexpr int std::numeric_limits< int >::denorm_min ()  [inline], [static], [constexpr],
[noexcept]
```

The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

epsilon()

```
static constexpr int std::numeric_limits< int >::epsilon ()  [inline], [static], [constexpr],
[noexcept]
```

The *machine epsilon:* the difference between 1 and the least value greater than 1 that is representable.

infinity()

static constexpr int std::numeric_limits< int >::infinity () [inline], [static], [constexpr], [noexcept]
The representation of positive infinity, if has_infinity.

lowest()

static constexpr int std::numeric_limits< int >::lowest () [inline], [static], [constexpr], [noexcept]
A finite value x such that there is no other finite value y where y < x.

max()

static constexpr int std::numeric_limits< int >::max () [inline], [static], [constexpr], [noexcept]
The maximum finite value.

min()

static constexpr int std::numeric_limits< int >::min () [inline], [static], [constexpr], [noexcept]
The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

quiet_NaN()

static constexpr int std::numeric_limits< int >::quiet_NaN () [inline], [static], [constexpr], [noexcept]
The representation of a quiet Not a Number, if has_quiet_NaN.

round_error()

static constexpr int std::numeric_limits< int >::round_error () [inline], [static], [constexpr], [noexcept]
The maximum rounding error measurement (see LIA-1).

signaling_NaN()

static constexpr int std::numeric_limits< int >::signaling_NaN () [inline], [static], [constexpr], [noexcept]
The representation of a signaling Not a Number, if has_signaling_NaN.

5.717.3 Member Data Documentation

digits

int std::__numeric_limits_base::digits [static], [constexpr]
The number of radix digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of radix digits in the mantissa.

digits10

int std::__numeric_limits_base::digits10 [static], [constexpr]
The number of base 10 digits that can be represented without change.

has_denorm

float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
See std::float_denorm_style for more information.

has_denorm_loss

```
bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
```
True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

has_infinity

```
bool std::__numeric_limits_base::has_infinity [static], [constexpr]
```
True if the type has a representation for positive infinity.

has_quiet_NaN

```
bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
```
True if the type has a representation for a quiet (non-signaling) Not a Number.

has_signaling_NaN

```
bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
```
True if the type has a representation for a signaling Not a Number.

is_bounded

```
bool std::__numeric_limits_base::is_bounded [static], [constexpr]
```
True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

is_exact

```
bool std::__numeric_limits_base::is_exact [static], [constexpr]
```
True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

is_iec559

```
bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
```
True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

is_integer

```
bool std::__numeric_limits_base::is_integer [static], [constexpr]
```
True if the type is integer.

is_modulo

```
bool std::__numeric_limits_base::is_modulo [static], [constexpr]
```
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or $*$ on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

is_signed

```
bool std::__numeric_limits_base::is_signed [static], [constexpr]
```
True if the type is signed.

is_specialized

`bool std::__numeric_limits_base::is_specialized [static], [constexpr]`
This will be true for all fundamental types (which have specializations), and false for everything else.

max_digits10

`int std::__numeric_limits_base::max_digits10 [static], [constexpr]`
The number of base 10 digits required to ensure that values which differ are always differentiated.

max_exponent

`int std::__numeric_limits_base::max_exponent [static], [constexpr]`
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

max_exponent10

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr]`
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

min_exponent

`int std::__numeric_limits_base::min_exponent [static], [constexpr]`
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

min_exponent10

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr]`
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

radix

`int std::__numeric_limits_base::radix [static], [constexpr]`
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

round_style

`float_round_style std::__numeric_limits_base::round_style [static], [constexpr]`
See std::float_round_style for more information. This is only meaningful for floating types; integer types will all be round_toward_zero.

tinyness_before

`bool std::__numeric_limits_base::tinyness_before [static], [constexpr]`
True if tininess is detected before rounding. (see IEC 559)

traps

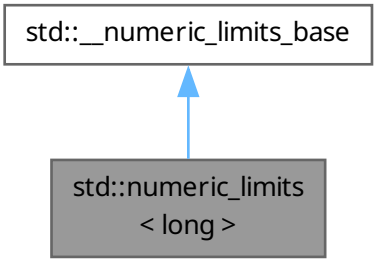
`bool std::__numeric_limits_base::traps [static], [constexpr]`
True if trapping is implemented for this type.
The documentation for this struct was generated from the following file:

- limits

5.718 std::numeric_limits< long > Struct Reference

`#include <limits>`

Inheritance diagram for std::numeric_limits< long >:



Static Public Member Functions

- static constexpr long denorm_min () noexcept
- static constexpr long **denorm_min** () noexcept
- static constexpr long epsilon () noexcept
- static constexpr long **epsilon** () noexcept
- static constexpr long infinity () noexcept
- static constexpr long **infinity** () noexcept
- static constexpr long lowest () noexcept
- static constexpr long **lowest** () noexcept
- static constexpr long max () noexcept
- static constexpr long **max** () noexcept
- static constexpr long min () noexcept
- static constexpr long **min** () noexcept
- static constexpr long quiet_NaN () noexcept
- static constexpr long **quiet_NaN** () noexcept
- static constexpr long round_error () noexcept
- static constexpr long **round_error** () noexcept
- static constexpr long signaling_NaN () noexcept
- static constexpr long **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int digits
- static constexpr int **digits**
- static constexpr int digits10
- static constexpr int **digits10**
- static constexpr float_denorm_style has_denorm
- static constexpr float_denorm_style **has_denorm**
- static constexpr bool has_denorm_loss
- static constexpr bool **has_denorm_loss**

- static constexpr bool has_infinity
- static constexpr bool **has_infinity**
- static constexpr bool has_quiet_NaN
- static constexpr bool **has_quiet_NaN**
- static constexpr bool has_signaling_NaN
- static constexpr bool **has_signaling_NaN**
- static constexpr bool is_bounded
- static constexpr bool **is_bounded**
- static constexpr bool is_exact
- static constexpr bool **is_exact**
- static constexpr bool is_iec559
- static constexpr bool **is_iec559**
- static constexpr bool is_integer
- static constexpr bool **is_integer**
- static constexpr bool is_modulo
- static constexpr bool **is_modulo**
- static constexpr bool is_signed
- static constexpr bool **is_signed**
- static constexpr bool is_specialized
- static constexpr bool **is_specialized**
- static constexpr int max_digits10
- static constexpr int **max_digits10**
- static constexpr int max_exponent
- static constexpr int **max_exponent**
- static constexpr int max_exponent10
- static constexpr int **max_exponent10**
- static constexpr int min_exponent
- static constexpr int **min_exponent**
- static constexpr int min_exponent10
- static constexpr int **min_exponent10**
- static constexpr int radix
- static constexpr int **radix**
- static constexpr float_round_style round_style
- static constexpr float_round_style **round_style**
- static constexpr bool tinyness_before
- static constexpr bool **tinyness_before**
- static constexpr bool traps
- static constexpr bool **traps**

5.718.1 Detailed Description

numeric_limits<long> specialization.

5.718.2 Member Function Documentation

denorm_min()

```
static constexpr long std::numeric_limits< long >::denorm_min ()  [inline], [static], [constexpr], [noexcept]
```

The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

epsilon()

static constexpr long std::numeric_limits< long >::epsilon () [inline], [static], [constexpr], [noexcept]
The *machine epsilon:* the difference between 1 and the least value greater than 1 that is representable.

infinity()

static constexpr long std::numeric_limits< long >::infinity () [inline], [static], [constexpr], [noexcept]
The representation of positive infinity, if `has_infinity`.

lowest()

static constexpr long std::numeric_limits< long >::lowest () [inline], [static], [constexpr], [noexcept]
A finite value x such that there is no other finite value y where y < x.

max()

static constexpr long std::numeric_limits< long >::max () [inline], [static], [constexpr], [noexcept]
The maximum finite value.

min()

static constexpr long std::numeric_limits< long >::min () [inline], [static], [constexpr], [noexcept]
The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

quiet_NaN()

static constexpr long std::numeric_limits< long >::quiet_NaN () [inline], [static], [constexpr], [noexcept]
The representation of a quiet Not a Number, if `has_quiet_NaN`.

round_error()

static constexpr long std::numeric_limits< long >::round_error () [inline], [static], [constexpr], [noexcept]
The maximum rounding error measurement (see LIA-1).

signaling_NaN()

static constexpr long std::numeric_limits< long >::signaling_NaN () [inline], [static], [constexpr], [noexcept]
The representation of a signaling Not a Number, if `has_signaling_NaN`.

5.718.3 Member Data Documentation

digits

int std::__numeric_limits_base::digits [static], [constexpr]
The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

digits10

`int std::__numeric_limits_base::digits10 [static], [constexpr]`
The number of base 10 digits that can be represented without change.

has_denorm

`float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]`
See std::float_denorm_style for more information.

has_denorm_loss

`bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]`
True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

has_infinity

`bool std::__numeric_limits_base::has_infinity [static], [constexpr]`
True if the type has a representation for positive infinity.

has_quiet_NaN

`bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]`
True if the type has a representation for a quiet (non-signaling) Not a Number.

has_signaling_NaN

`bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]`
True if the type has a representation for a signaling Not a Number.

is_bounded

`bool std::__numeric_limits_base::is_bounded [static], [constexpr]`
True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

is_exact

`bool std::__numeric_limits_base::is_exact [static], [constexpr]`
True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

is_iec559

`bool std::__numeric_limits_base::is_iec559 [static], [constexpr]`
True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

is_integer

`bool std::__numeric_limits_base::is_integer [static], [constexpr]`
True if the type is integer.

is_modulo

`bool std::__numeric_limits_base::is_modulo [static], [constexpr]`
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or ∗ on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

is_signed

`bool std::__numeric_limits_base::is_signed [static], [constexpr]`
True if the type is signed.

is_specialized

`bool std::__numeric_limits_base::is_specialized [static], [constexpr]`
This will be true for all fundamental types (which have specializations), and false for everything else.

max_digits10

`int std::__numeric_limits_base::max_digits10 [static], [constexpr]`
The number of base 10 digits required to ensure that values which differ are always differentiated.

max_exponent

`int std::__numeric_limits_base::max_exponent [static], [constexpr]`
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

max_exponent10

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr]`
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

min_exponent

`int std::__numeric_limits_base::min_exponent [static], [constexpr]`
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

min_exponent10

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr]`
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

radix

`int std::__numeric_limits_base::radix [static], [constexpr]`
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

round_style

`float_round_style std::__numeric_limits_base::round_style [static], [constexpr]`
See std::float_round_style for more information. This is only meaningful for floating types; integer types will all be round_toward_zero.

tinyness_before

`bool std::__numeric_limits_base::tinyness_before [static], [constexpr]`
True if tininess is detected before rounding. (see IEC 559)

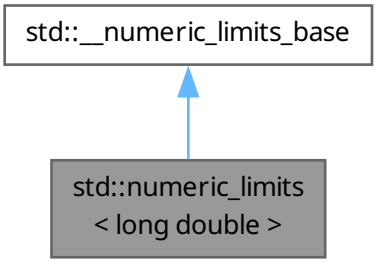
traps

`bool std::__numeric_limits_base::traps [static], [constexpr]`
True if trapping is implemented for this type.
The documentation for this struct was generated from the following file:

- limits

5.719 std::numeric_limits< long double > Struct Reference

`#include <limits>`
Inheritance diagram for std::numeric_limits< long double >:



Static Public Member Functions

- static constexpr long double denorm_min () noexcept
- static constexpr long double **denorm_min** () noexcept
- static constexpr long double epsilon () noexcept
- static constexpr long double **epsilon** () noexcept
- static constexpr long double infinity () noexcept
- static constexpr long double **infinity** () noexcept
- static constexpr long double lowest () noexcept
- static constexpr long double **lowest** () noexcept
- static constexpr long double max () noexcept
- static constexpr long double **max** () noexcept
- static constexpr long double min () noexcept
- static constexpr long double **min** () noexcept

- static constexpr long double quiet_NaN () noexcept
- static constexpr long double **quiet_NaN** () noexcept
- static constexpr long double round_error () noexcept
- static constexpr long double **round_error** () noexcept
- static constexpr long double signaling_NaN () noexcept
- static constexpr long double **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int digits
- static constexpr int **digits**
- static constexpr int digits10
- static constexpr int **digits10**
- static constexpr float_denorm_style has_denorm
- static constexpr float_denorm_style **has_denorm**
- static constexpr bool has_denorm_loss
- static constexpr bool **has_denorm_loss**
- static constexpr bool has_infinity
- static constexpr bool **has_infinity**
- static constexpr bool has_quiet_NaN
- static constexpr bool **has_quiet_NaN**
- static constexpr bool has_signaling_NaN
- static constexpr bool **has_signaling_NaN**
- static constexpr bool is_bounded
- static constexpr bool **is_bounded**
- static constexpr bool is_exact
- static constexpr bool **is_exact**
- static constexpr bool is_iec559
- static constexpr bool **is_iec559**
- static constexpr bool is_integer
- static constexpr bool **is_integer**
- static constexpr bool is_modulo
- static constexpr bool **is_modulo**
- static constexpr bool is_signed
- static constexpr bool **is_signed**
- static constexpr bool is_specialized
- static constexpr bool **is_specialized**
- static constexpr int max_digits10
- static constexpr int **max_digits10**
- static constexpr int max_exponent
- static constexpr int **max_exponent**
- static constexpr int max_exponent10
- static constexpr int **max_exponent10**
- static constexpr int min_exponent
- static constexpr int **min_exponent**
- static constexpr int min_exponent10
- static constexpr int **min_exponent10**
- static constexpr int radix
- static constexpr int **radix**
- static constexpr float_round_style round_style
- static constexpr float_round_style **round_style**

- static constexpr bool tinyness_before
- static constexpr bool **tinyness_before**
- static constexpr bool traps
- static constexpr bool **traps**

5.719.1 Detailed Description

numeric_limits<long double> specialization.

5.719.2 Member Function Documentation

denorm_min()

```
static constexpr long double std::numeric_limits< long double >::denorm_min ()  [inline], [static],
[constexpr], [noexcept]
```

The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

epsilon()

```
static constexpr long double std::numeric_limits< long double >::epsilon ()  [inline], [static],
[constexpr], [noexcept]
```

The *machine epsilon:* the difference between 1 and the least value greater than 1 that is representable.

infinity()

```
static constexpr long double std::numeric_limits< long double >::infinity ()  [inline], [static],
[constexpr], [noexcept]
```

The representation of positive infinity, if `has_infinity`.

lowest()

```
static constexpr long double std::numeric_limits< long double >::lowest ()  [inline], [static],
[constexpr], [noexcept]
```

A finite value x such that there is no other finite value y where y < x.

max()

```
static constexpr long double std::numeric_limits< long double >::max ()  [inline], [static],
[constexpr], [noexcept]
```

The maximum finite value.

min()

```
static constexpr long double std::numeric_limits< long double >::min ()  [inline], [static],
[constexpr], [noexcept]
```

The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

quiet_NaN()

```
static constexpr long double std::numeric_limits< long double >::quiet_NaN ()  [inline], [static],
[constexpr], [noexcept]
```

The representation of a quiet Not a Number, if `has_quiet_NaN`.

round_error()

static constexpr long double std::numeric_limits< long double >::round_error () [inline], [static], [constexpr], [noexcept]
The maximum rounding error measurement (see LIA-1).

signaling_NaN()

static constexpr long double std::numeric_limits< long double >::signaling_NaN () [inline], [static], [constexpr], [noexcept]
The representation of a signaling Not a Number, if has_signaling_NaN.

5.719.3 Member Data Documentation

digits

int std::__numeric_limits_base::digits [static], [constexpr]
The number of radix digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of radix digits in the mantissa.

digits10

int std::__numeric_limits_base::digits10 [static], [constexpr]
The number of base 10 digits that can be represented without change.

has_denorm

float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
See std::float_denorm_style for more information.

has_denorm_loss

bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

has_infinity

bool std::__numeric_limits_base::has_infinity [static], [constexpr]
True if the type has a representation for positive infinity.

has_quiet_NaN

bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
True if the type has a representation for a quiet (non-signaling) Not a Number.

has_signaling_NaN

bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
True if the type has a representation for a signaling Not a Number.

is_bounded

bool std::__numeric_limits_base::is_bounded [static], [constexpr]
True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

is_exact

`bool std::__numeric_limits_base::is_exact [static], [constexpr]`
True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

is_iec559

`bool std::__numeric_limits_base::is_iec559 [static], [constexpr]`
True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

is_integer

`bool std::__numeric_limits_base::is_integer [static], [constexpr]`
True if the type is integer.

is_modulo

`bool std::__numeric_limits_base::is_modulo [static], [constexpr]`
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or $*$ on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

is_signed

`bool std::__numeric_limits_base::is_signed [static], [constexpr]`
True if the type is signed.

is_specialized

`bool std::__numeric_limits_base::is_specialized [static], [constexpr]`
This will be true for all fundamental types (which have specializations), and false for everything else.

max_digits10

`int std::__numeric_limits_base::max_digits10 [static], [constexpr]`
The number of base 10 digits required to ensure that values which differ are always differentiated.

max_exponent

`int std::__numeric_limits_base::max_exponent [static], [constexpr]`
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

max_exponent10

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr]`
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

min_exponent

`int std::__numeric_limits_base::min_exponent [static], [constexpr]`
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

min_exponent10

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr]`
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

radix

`int std::__numeric_limits_base::radix [static], [constexpr]`
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

round_style

`float_round_style std::__numeric_limits_base::round_style [static], [constexpr]`
See std::float_round_style for more information. This is only meaningful for floating types; integer types will all be round_toward_zero.

tinyness_before

`bool std::__numeric_limits_base::tinyness_before [static], [constexpr]`
True if tininess is detected before rounding. (see IEC 559)

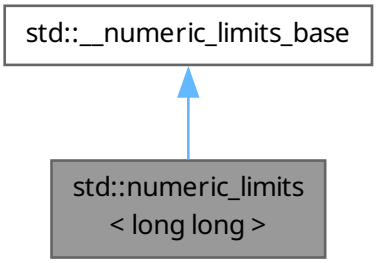
traps

`bool std::__numeric_limits_base::traps [static], [constexpr]`
True if trapping is implemented for this type.
The documentation for this struct was generated from the following file:

- limits

5.720 std::numeric_limits< long long > Struct Reference

`#include <limits>`
Inheritance diagram for std::numeric_limits< long long >:



Static Public Member Functions

- static constexpr long long denorm_min () noexcept
- static constexpr long long **denorm_min** () noexcept
- static constexpr long long epsilon () noexcept
- static constexpr long long **epsilon** () noexcept
- static constexpr long long infinity () noexcept
- static constexpr long long **infinity** () noexcept
- static constexpr long long lowest () noexcept
- static constexpr long long **lowest** () noexcept
- static constexpr long long max () noexcept
- static constexpr long long **max** () noexcept
- static constexpr long long min () noexcept
- static constexpr long long **min** () noexcept
- static constexpr long long quiet_NaN () noexcept
- static constexpr long long **quiet_NaN** () noexcept
- static constexpr long long round_error () noexcept
- static constexpr long long **round_error** () noexcept
- static constexpr long long signaling_NaN () noexcept
- static constexpr long long **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int digits
- static constexpr int **digits**
- static constexpr int digits10
- static constexpr int **digits10**
- static constexpr float_denorm_style has_denorm
- static constexpr float_denorm_style **has_denorm**
- static constexpr bool has_denorm_loss
- static constexpr bool **has_denorm_loss**
- static constexpr bool has_infinity
- static constexpr bool **has_infinity**
- static constexpr bool has_quiet_NaN
- static constexpr bool **has_quiet_NaN**
- static constexpr bool has_signaling_NaN
- static constexpr bool **has_signaling_NaN**
- static constexpr bool is_bounded
- static constexpr bool **is_bounded**
- static constexpr bool is_exact
- static constexpr bool **is_exact**
- static constexpr bool is_iec559
- static constexpr bool **is_iec559**
- static constexpr bool is_integer
- static constexpr bool **is_integer**
- static constexpr bool is_modulo
- static constexpr bool **is_modulo**
- static constexpr bool is_signed
- static constexpr bool **is_signed**
- static constexpr bool is_specialized
- static constexpr bool **is_specialized**
- static constexpr int max_digits10

- static constexpr int **max_digits10**
- static constexpr int max_exponent
- static constexpr int **max_exponent**
- static constexpr int max_exponent10
- static constexpr int **max_exponent10**
- static constexpr int min_exponent
- static constexpr int **min_exponent**
- static constexpr int min_exponent10
- static constexpr int **min_exponent10**
- static constexpr int radix
- static constexpr int **radix**
- static constexpr float_round_style round_style
- static constexpr float_round_style **round_style**
- static constexpr bool tinyness_before
- static constexpr bool **tinyness_before**
- static constexpr bool traps
- static constexpr bool **traps**

5.720.1 Detailed Description

numeric_limits<long long> specialization.

5.720.2 Member Function Documentation

denorm_min()

```
static constexpr long long std::numeric_limits< long long >::denorm_min ()  [inline], [static],
[constexpr], [noexcept]
```

The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

epsilon()

```
static constexpr long long std::numeric_limits< long long >::epsilon ()  [inline], [static],
[constexpr], [noexcept]
```

The *machine epsilon:* the difference between 1 and the least value greater than 1 that is representable.

infinity()

```
static constexpr long long std::numeric_limits< long long >::infinity ()  [inline], [static],
[constexpr], [noexcept]
```

The representation of positive infinity, if `has_infinity`.

lowest()

```
static constexpr long long std::numeric_limits< long long >::lowest ()  [inline], [static], [constexpr],
[noexcept]
```

A finite value x such that there is no other finite value y where y < x.

max()

```
static constexpr long long std::numeric_limits< long long >::max ()  [inline], [static], [constexpr],
[noexcept]
```

The maximum finite value.

min()

`static constexpr long long std::numeric_limits< long long >::min () [inline], [static], [constexpr], [noexcept]`
The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

quiet_NaN()

`static constexpr long long std::numeric_limits< long long >::quiet_NaN () [inline], [static], [constexpr], [noexcept]`
The representation of a quiet Not a Number, if `has_quiet_NaN`.

round_error()

`static constexpr long long std::numeric_limits< long long >::round_error () [inline], [static], [constexpr], [noexcept]`
The maximum rounding error measurement (see LIA-1).

signaling_NaN()

`static constexpr long long std::numeric_limits< long long >::signaling_NaN () [inline], [static], [constexpr], [noexcept]`
The representation of a signaling Not a Number, if `has_signaling_NaN`.

5.720.3 Member Data Documentation

digits

`int std::__numeric_limits_base::digits [static], [constexpr]`
The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

digits10

`int std::__numeric_limits_base::digits10 [static], [constexpr]`
The number of base 10 digits that can be represented without change.

has_denorm

`float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]`
See std::float_denorm_style for more information.

has_denorm_loss

`bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]`
True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

has_infinity

`bool std::__numeric_limits_base::has_infinity [static], [constexpr]`
True if the type has a representation for positive infinity.

has_quiet_NaN

`bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]`
True if the type has a representation for a quiet (non-signaling) Not a Number.

has_signaling_NaN

bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
True if the type has a representation for a signaling Not a Number.

is_bounded

bool std::__numeric_limits_base::is_bounded [static], [constexpr]
True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

is_exact

bool std::__numeric_limits_base::is_exact [static], [constexpr]
True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

is_iec559

bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

is_integer

bool std::__numeric_limits_base::is_integer [static], [constexpr]
True if the type is integer.

is_modulo

bool std::__numeric_limits_base::is_modulo [static], [constexpr]
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or $*$ on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

is_signed

bool std::__numeric_limits_base::is_signed [static], [constexpr]
True if the type is signed.

is_specialized

bool std::__numeric_limits_base::is_specialized [static], [constexpr]
This will be true for all fundamental types (which have specializations), and false for everything else.

max_digits10

int std::__numeric_limits_base::max_digits10 [static], [constexpr]
The number of base 10 digits required to ensure that values which differ are always differentiated.

max_exponent

int std::__numeric_limits_base::max_exponent [static], [constexpr]
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

max_exponent10

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr]`
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

min_exponent

`int std::__numeric_limits_base::min_exponent [static], [constexpr]`
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

min_exponent10

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr]`
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

radix

`int std::__numeric_limits_base::radix [static], [constexpr]`
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

round_style

`float_round_style std::__numeric_limits_base::round_style [static], [constexpr]`
See std::float_round_style for more information. This is only meaningful for floating types; integer types will all be round_toward_zero.

tinyness_before

`bool std::__numeric_limits_base::tinyness_before [static], [constexpr]`
True if tininess is detected before rounding. (see IEC 559)

traps

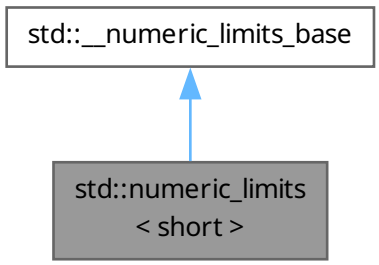
`bool std::__numeric_limits_base::traps [static], [constexpr]`
True if trapping is implemented for this type.
The documentation for this struct was generated from the following file:

- limits

5.721 std::numeric_limits< short > Struct Reference

`#include <limits>`

Inheritance diagram for std::numeric_limits< short >:



Static Public Member Functions

- static constexpr short denorm_min () noexcept
- static constexpr short **denorm_min** () noexcept
- static constexpr short epsilon () noexcept
- static constexpr short **epsilon** () noexcept
- static constexpr short infinity () noexcept
- static constexpr short **infinity** () noexcept
- static constexpr short lowest () noexcept
- static constexpr short **lowest** () noexcept
- static constexpr short max () noexcept
- static constexpr short **max** () noexcept
- static constexpr short min () noexcept
- static constexpr short **min** () noexcept
- static constexpr short quiet_NaN () noexcept
- static constexpr short **quiet_NaN** () noexcept
- static constexpr short round_error () noexcept
- static constexpr short **round_error** () noexcept
- static constexpr short signaling_NaN () noexcept
- static constexpr short **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int digits
- static constexpr int **digits**
- static constexpr int digits10
- static constexpr int **digits10**
- static constexpr float_denorm_style has_denorm
- static constexpr float_denorm_style **has_denorm**
- static constexpr bool has_denorm_loss
- static constexpr bool **has_denorm_loss**
- static constexpr bool has_infinity
- static constexpr bool **has_infinity**
- static constexpr bool has_quiet_NaN

- static constexpr bool **has_quiet_NaN**
- static constexpr bool has_signaling_NaN
- static constexpr bool **has_signaling_NaN**
- static constexpr bool is_bounded
- static constexpr bool **is_bounded**
- static constexpr bool is_exact
- static constexpr bool **is_exact**
- static constexpr bool is_iec559
- static constexpr bool **is_iec559**
- static constexpr bool is_integer
- static constexpr bool **is_integer**
- static constexpr bool is_modulo
- static constexpr bool **is_modulo**
- static constexpr bool is_signed
- static constexpr bool **is_signed**
- static constexpr bool is_specialized
- static constexpr bool **is_specialized**
- static constexpr int max_digits10
- static constexpr int **max_digits10**
- static constexpr int max_exponent
- static constexpr int **max_exponent**
- static constexpr int max_exponent10
- static constexpr int **max_exponent10**
- static constexpr int min_exponent
- static constexpr int **min_exponent**
- static constexpr int min_exponent10
- static constexpr int **min_exponent10**
- static constexpr int radix
- static constexpr int **radix**
- static constexpr float_round_style round_style
- static constexpr float_round_style **round_style**
- static constexpr bool tinyness_before
- static constexpr bool **tinyness_before**
- static constexpr bool traps
- static constexpr bool **traps**

5.721.1 Detailed Description

numeric_limits<short> specialization.

5.721.2 Member Function Documentation

denorm_min()

```
static constexpr short std::numeric_limits< short >::denorm_min ()  [inline], [static], [constexpr],
[noexcept]
```

The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

epsilon()

```
static constexpr short std::numeric_limits< short >::epsilon ()  [inline], [static], [constexpr],
[noexcept]
```

The *machine epsilon:* the difference between 1 and the least value greater than 1 that is representable.

infinity()

```
static constexpr short std::numeric_limits< short >::infinity ()  [inline], [static], [constexpr], [noexcept]
```

The representation of positive infinity, if `has_infinity`.

lowest()

```
static constexpr short std::numeric_limits< short >::lowest ()  [inline], [static], [constexpr], [noexcept]
```

A finite value x such that there is no other finite value y where y < x.

max()

```
static constexpr short std::numeric_limits< short >::max ()  [inline], [static], [constexpr], [noexcept]
```

The maximum finite value.

min()

```
static constexpr short std::numeric_limits< short >::min ()  [inline], [static], [constexpr], [noexcept]
```

The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

quiet_NaN()

```
static constexpr short std::numeric_limits< short >::quiet_NaN ()  [inline], [static], [constexpr], [noexcept]
```

The representation of a quiet Not a Number, if `has_quiet_NaN`.

round_error()

```
static constexpr short std::numeric_limits< short >::round_error ()  [inline], [static], [constexpr], [noexcept]
```

The maximum rounding error measurement (see LIA-1).

signaling_NaN()

```
static constexpr short std::numeric_limits< short >::signaling_NaN ()  [inline], [static], [constexpr], [noexcept]
```

The representation of a signaling Not a Number, if `has_signaling_NaN`.

5.721.3 Member Data Documentation

digits

```
int std::__numeric_limits_base::digits  [static], [constexpr]
```

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

digits10

```
int std::__numeric_limits_base::digits10  [static], [constexpr]
```

The number of base 10 digits that can be represented without change.

has_denorm

`float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]`
See std::float_denorm_style for more information.

has_denorm_loss

`bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]`
True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

has_infinity

`bool std::__numeric_limits_base::has_infinity [static], [constexpr]`
True if the type has a representation for positive infinity.

has_quiet_NaN

`bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]`
True if the type has a representation for a quiet (non-signaling) Not a Number.

has_signaling_NaN

`bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]`
True if the type has a representation for a signaling Not a Number.

is_bounded

`bool std::__numeric_limits_base::is_bounded [static], [constexpr]`
True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

is_exact

`bool std::__numeric_limits_base::is_exact [static], [constexpr]`
True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

is_iec559

`bool std::__numeric_limits_base::is_iec559 [static], [constexpr]`
True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

is_integer

`bool std::__numeric_limits_base::is_integer [static], [constexpr]`
True if the type is integer.

is_modulo

`bool std::__numeric_limits_base::is_modulo [static], [constexpr]`
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or $*$ on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

is_signed

bool std::__numeric_limits_base::is_signed [static], [constexpr]
True if the type is signed.

is_specialized

bool std::__numeric_limits_base::is_specialized [static], [constexpr]
This will be true for all fundamental types (which have specializations), and false for everything else.

max_digits10

int std::__numeric_limits_base::max_digits10 [static], [constexpr]
The number of base 10 digits required to ensure that values which differ are always differentiated.

max_exponent

int std::__numeric_limits_base::max_exponent [static], [constexpr]
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

max_exponent10

int std::__numeric_limits_base::max_exponent10 [static], [constexpr]
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

min_exponent

int std::__numeric_limits_base::min_exponent [static], [constexpr]
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

min_exponent10

int std::__numeric_limits_base::min_exponent10 [static], [constexpr]
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

radix

int std::__numeric_limits_base::radix [static], [constexpr]
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

round_style

float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
See std::float_round_style for more information. This is only meaningful for floating types; integer types will all be round_toward_zero.

tinyness_before

bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
True if tininess is detected before rounding. (see IEC 559)

traps

```
bool std::__numeric_limits_base::traps  [static], [constexpr]
```

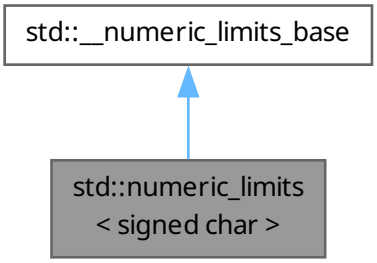
True if trapping is implemented for this type.
The documentation for this struct was generated from the following file:

- limits

5.722 std::numeric_limits< signed char > Struct Reference

```
#include <limits>
```

Inheritance diagram for std::numeric_limits< signed char >:



Static Public Member Functions

- static constexpr signed char denorm_min () noexcept
- static constexpr signed char **denorm_min** () noexcept
- static constexpr signed char epsilon () noexcept
- static constexpr signed char **epsilon** () noexcept
- static constexpr signed char infinity () noexcept
- static constexpr signed char **infinity** () noexcept
- static constexpr signed char lowest () noexcept
- static constexpr signed char **lowest** () noexcept
- static constexpr signed char max () noexcept
- static constexpr signed char **max** () noexcept
- static constexpr signed char min () noexcept
- static constexpr signed char **min** () noexcept
- static constexpr signed char quiet_NaN () noexcept
- static constexpr signed char **quiet_NaN** () noexcept
- static constexpr signed char round_error () noexcept
- static constexpr signed char **round_error** () noexcept
- static constexpr signed char signaling_NaN () noexcept
- static constexpr signed char **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int digits
- static constexpr int **digits**
- static constexpr int digits10
- static constexpr int **digits10**
- static constexpr float_denorm_style has_denorm
- static constexpr float_denorm_style **has_denorm**
- static constexpr bool has_denorm_loss
- static constexpr bool **has_denorm_loss**
- static constexpr bool has_infinity
- static constexpr bool **has_infinity**
- static constexpr bool has_quiet_NaN
- static constexpr bool **has_quiet_NaN**
- static constexpr bool has_signaling_NaN
- static constexpr bool **has_signaling_NaN**
- static constexpr bool is_bounded
- static constexpr bool **is_bounded**
- static constexpr bool is_exact
- static constexpr bool **is_exact**
- static constexpr bool is_iec559
- static constexpr bool **is_iec559**
- static constexpr bool is_integer
- static constexpr bool **is_integer**
- static constexpr bool is_modulo
- static constexpr bool **is_modulo**
- static constexpr bool is_signed
- static constexpr bool **is_signed**
- static constexpr bool is_specialized
- static constexpr bool **is_specialized**
- static constexpr int max_digits10
- static constexpr int **max_digits10**
- static constexpr int max_exponent
- static constexpr int **max_exponent**
- static constexpr int max_exponent10
- static constexpr int **max_exponent10**
- static constexpr int min_exponent
- static constexpr int **min_exponent**
- static constexpr int min_exponent10
- static constexpr int **min_exponent10**
- static constexpr int radix
- static constexpr int **radix**
- static constexpr float_round_style round_style
- static constexpr float_round_style **round_style**
- static constexpr bool tinyness_before
- static constexpr bool **tinyness_before**
- static constexpr bool traps
- static constexpr bool **traps**

5.722.1 Detailed Description

numeric_limits<signed char> specialization.

5.722.2 Member Function Documentation

denorm_min()

`static constexpr signed char std::numeric_limits< signed char >::denorm_min () [inline], [static], [constexpr], [noexcept]`

The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

epsilon()

`static constexpr signed char std::numeric_limits< signed char >::epsilon () [inline], [static], [constexpr], [noexcept]`

The *machine epsilon:* the difference between 1 and the least value greater than 1 that is representable.

infinity()

`static constexpr signed char std::numeric_limits< signed char >::infinity () [inline], [static], [constexpr], [noexcept]`

The representation of positive infinity, if `has_infinity`.

lowest()

`static constexpr signed char std::numeric_limits< signed char >::lowest () [inline], [static], [constexpr], [noexcept]`

A finite value x such that there is no other finite value y where y $<$ x.

max()

`static constexpr signed char std::numeric_limits< signed char >::max () [inline], [static], [constexpr], [noexcept]`

The maximum finite value.

min()

`static constexpr signed char std::numeric_limits< signed char >::min () [inline], [static], [constexpr], [noexcept]`

The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

quiet_NaN()

`static constexpr signed char std::numeric_limits< signed char >::quiet_NaN () [inline], [static], [constexpr], [noexcept]`

The representation of a quiet Not a Number, if `has_quiet_NaN`.

round_error()

`static constexpr signed char std::numeric_limits< signed char >::round_error () [inline], [static], [constexpr], [noexcept]`

The maximum rounding error measurement (see LIA-1).

signaling_NaN()

`static constexpr signed char std::numeric_limits< signed char >::signaling_NaN () [inline], [static], [constexpr], [noexcept]`

The representation of a signaling Not a Number, if `has_signaling_NaN`.

5.722.3 Member Data Documentation

digits

```
int std::__numeric_limits_base::digits  [static], [constexpr]
```
The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

digits10

```
int std::__numeric_limits_base::digits10  [static], [constexpr]
```
The number of base 10 digits that can be represented without change.

has_denorm

```
float_denorm_style std::__numeric_limits_base::has_denorm  [static], [constexpr]
```
See std::float_denorm_style for more information.

has_denorm_loss

```
bool std::__numeric_limits_base::has_denorm_loss  [static], [constexpr]
```
True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

has_infinity

```
bool std::__numeric_limits_base::has_infinity  [static], [constexpr]
```
True if the type has a representation for positive infinity.

has_quiet_NaN

```
bool std::__numeric_limits_base::has_quiet_NaN  [static], [constexpr]
```
True if the type has a representation for a quiet (non-signaling) Not a Number.

has_signaling_NaN

```
bool std::__numeric_limits_base::has_signaling_NaN  [static], [constexpr]
```
True if the type has a representation for a signaling Not a Number.

is_bounded

```
bool std::__numeric_limits_base::is_bounded  [static], [constexpr]
```
True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

is_exact

```
bool std::__numeric_limits_base::is_exact  [static], [constexpr]
```
True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

is_iec559

```
bool std::__numeric_limits_base::is_iec559  [static], [constexpr]
```
True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

is_integer

`bool std::__numeric_limits_base::is_integer [static], [constexpr]`
True if the type is integer.

is_modulo

`bool std::__numeric_limits_base::is_modulo [static], [constexpr]`
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or $*$ on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

is_signed

`bool std::__numeric_limits_base::is_signed [static], [constexpr]`
True if the type is signed.

is_specialized

`bool std::__numeric_limits_base::is_specialized [static], [constexpr]`
This will be true for all fundamental types (which have specializations), and false for everything else.

max_digits10

`int std::__numeric_limits_base::max_digits10 [static], [constexpr]`
The number of base 10 digits required to ensure that values which differ are always differentiated.

max_exponent

`int std::__numeric_limits_base::max_exponent [static], [constexpr]`
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

max_exponent10

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr]`
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

min_exponent

`int std::__numeric_limits_base::min_exponent [static], [constexpr]`
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

min_exponent10

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr]`
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

radix

`int std::__numeric_limits_base::radix [static], [constexpr]`
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

round_style

```
float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
```
See std::float_round_style for more information. This is only meaningful for floating types; integer types will all be round_toward_zero.

tinyness_before

```
bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
```
True if tininess is detected before rounding. (see IEC 559)

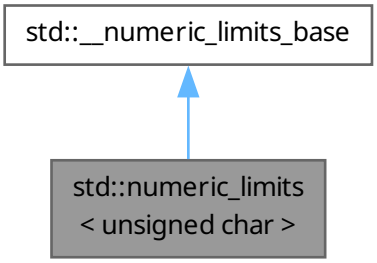
traps

```
bool std::__numeric_limits_base::traps [static], [constexpr]
```
True if trapping is implemented for this type.
The documentation for this struct was generated from the following file:

- limits

5.723 std::numeric_limits< unsigned char > Struct Reference

```
#include <limits>
```
Inheritance diagram for std::numeric_limits< unsigned char >:



Static Public Member Functions

- static constexpr unsigned char denorm_min () noexcept
- static constexpr unsigned char **denorm_min** () noexcept
- static constexpr unsigned char epsilon () noexcept
- static constexpr unsigned char **epsilon** () noexcept
- static constexpr unsigned char infinity () noexcept
- static constexpr unsigned char **infinity** () noexcept
- static constexpr unsigned char lowest () noexcept
- static constexpr unsigned char **lowest** () noexcept
- static constexpr unsigned char max () noexcept
- static constexpr unsigned char **max** () noexcept
- static constexpr unsigned char min () noexcept
- static constexpr unsigned char **min** () noexcept

- static constexpr unsigned char quiet_NaN () noexcept
- static constexpr unsigned char **quiet_NaN** () noexcept
- static constexpr unsigned char round_error () noexcept
- static constexpr unsigned char **round_error** () noexcept
- static constexpr unsigned char signaling_NaN () noexcept
- static constexpr unsigned char **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int digits
- static constexpr int **digits**
- static constexpr int digits10
- static constexpr int **digits10**
- static constexpr float_denorm_style has_denorm
- static constexpr float_denorm_style **has_denorm**
- static constexpr bool has_denorm_loss
- static constexpr bool **has_denorm_loss**
- static constexpr bool has_infinity
- static constexpr bool **has_infinity**
- static constexpr bool has_quiet_NaN
- static constexpr bool **has_quiet_NaN**
- static constexpr bool has_signaling_NaN
- static constexpr bool **has_signaling_NaN**
- static constexpr bool is_bounded
- static constexpr bool **is_bounded**
- static constexpr bool is_exact
- static constexpr bool **is_exact**
- static constexpr bool is_iec559
- static constexpr bool **is_iec559**
- static constexpr bool is_integer
- static constexpr bool **is_integer**
- static constexpr bool is_modulo
- static constexpr bool **is_modulo**
- static constexpr bool is_signed
- static constexpr bool **is_signed**
- static constexpr bool is_specialized
- static constexpr bool **is_specialized**
- static constexpr int max_digits10
- static constexpr int **max_digits10**
- static constexpr int max_exponent
- static constexpr int **max_exponent**
- static constexpr int max_exponent10
- static constexpr int **max_exponent10**
- static constexpr int min_exponent
- static constexpr int **min_exponent**
- static constexpr int min_exponent10
- static constexpr int **min_exponent10**
- static constexpr int radix
- static constexpr int **radix**
- static constexpr float_round_style round_style
- static constexpr float_round_style **round_style**

- static constexpr bool tinyness_before
- static constexpr bool **tinyness_before**
- static constexpr bool traps
- static constexpr bool **traps**

5.723.1 Detailed Description

numeric_limits<unsigned char> specialization.

5.723.2 Member Function Documentation

denorm_min()

```
static constexpr unsigned char std::numeric_limits< unsigned char >::denorm_min ()  [inline],
[static], [constexpr], [noexcept]
```

The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

epsilon()

```
static constexpr unsigned char std::numeric_limits< unsigned char >::epsilon ()  [inline], [static],
[constexpr], [noexcept]
```

The *machine epsilon:* the difference between 1 and the least value greater than 1 that is representable.

infinity()

```
static constexpr unsigned char std::numeric_limits< unsigned char >::infinity ()  [inline], [static],
[constexpr], [noexcept]
```

The representation of positive infinity, if `has_infinity`.

lowest()

```
static constexpr unsigned char std::numeric_limits< unsigned char >::lowest ()  [inline], [static],
[constexpr], [noexcept]
```

A finite value x such that there is no other finite value y where y < x.

max()

```
static constexpr unsigned char std::numeric_limits< unsigned char >::max ()  [inline], [static],
[constexpr], [noexcept]
```

The maximum finite value.

min()

```
static constexpr unsigned char std::numeric_limits< unsigned char >::min ()  [inline], [static],
[constexpr], [noexcept]
```

The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

quiet_NaN()

```
static constexpr unsigned char std::numeric_limits< unsigned char >::quiet_NaN ()  [inline],
[static], [constexpr], [noexcept]
```

The representation of a quiet Not a Number, if `has_quiet_NaN`.

round_error()

`static constexpr unsigned char std::numeric_limits< unsigned char >::round_error () [inline], [static], [constexpr], [noexcept]`
The maximum rounding error measurement (see LIA-1).

signaling_NaN()

`static constexpr unsigned char std::numeric_limits< unsigned char >::signaling_NaN () [inline], [static], [constexpr], [noexcept]`
The representation of a signaling Not a Number, if `has_signaling_NaN`.

5.723.3 Member Data Documentation

digits

`int std::__numeric_limits_base::digits [static], [constexpr]`
The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

digits10

`int std::__numeric_limits_base::digits10 [static], [constexpr]`
The number of base 10 digits that can be represented without change.

has_denorm

`float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]`
See std::float_denorm_style for more information.

has_denorm_loss

`bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]`
True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

has_infinity

`bool std::__numeric_limits_base::has_infinity [static], [constexpr]`
True if the type has a representation for positive infinity.

has_quiet_NaN

`bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]`
True if the type has a representation for a quiet (non-signaling) Not a Number.

has_signaling_NaN

`bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]`
True if the type has a representation for a signaling Not a Number.

is_bounded

`bool std::__numeric_limits_base::is_bounded [static], [constexpr]`
True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

is_exact

`bool std::__numeric_limits_base::is_exact [static], [constexpr]`
True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

is_iec559

`bool std::__numeric_limits_base::is_iec559 [static], [constexpr]`
True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

is_integer

`bool std::__numeric_limits_base::is_integer [static], [constexpr]`
True if the type is integer.

is_modulo

`bool std::__numeric_limits_base::is_modulo [static], [constexpr]`
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or $*$ on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

is_signed

`bool std::__numeric_limits_base::is_signed [static], [constexpr]`
True if the type is signed.

is_specialized

`bool std::__numeric_limits_base::is_specialized [static], [constexpr]`
This will be true for all fundamental types (which have specializations), and false for everything else.

max_digits10

`int std::__numeric_limits_base::max_digits10 [static], [constexpr]`
The number of base 10 digits required to ensure that values which differ are always differentiated.

max_exponent

`int std::__numeric_limits_base::max_exponent [static], [constexpr]`
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

max_exponent10

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr]`
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

min_exponent

`int std::__numeric_limits_base::min_exponent [static], [constexpr]`
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

min_exponent10

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr]`
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

radix

`int std::__numeric_limits_base::radix [static], [constexpr]`
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

round_style

`float_round_style std::__numeric_limits_base::round_style [static], [constexpr]`
See std::float_round_style for more information. This is only meaningful for floating types; integer types will all be round_toward_zero.

tinyness_before

`bool std::__numeric_limits_base::tinyness_before [static], [constexpr]`
True if tininess is detected before rounding. (see IEC 559)

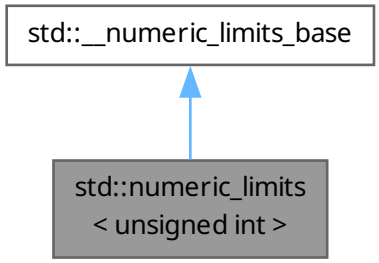
traps

`bool std::__numeric_limits_base::traps [static], [constexpr]`
True if trapping is implemented for this type.
The documentation for this struct was generated from the following file:

- limits

5.724 std::numeric_limits< unsigned int > Struct Reference

`#include <limits>`
Inheritance diagram for std::numeric_limits< unsigned int >:



Static Public Member Functions

- static constexpr unsigned int denorm_min () noexcept
- static constexpr unsigned int **denorm_min** () noexcept
- static constexpr unsigned int epsilon () noexcept
- static constexpr unsigned int **epsilon** () noexcept
- static constexpr unsigned int infinity () noexcept
- static constexpr unsigned int **infinity** () noexcept
- static constexpr unsigned int lowest () noexcept
- static constexpr unsigned int **lowest** () noexcept
- static constexpr unsigned int max () noexcept
- static constexpr unsigned int **max** () noexcept
- static constexpr unsigned int min () noexcept
- static constexpr unsigned int **min** () noexcept
- static constexpr unsigned int quiet_NaN () noexcept
- static constexpr unsigned int **quiet_NaN** () noexcept
- static constexpr unsigned int round_error () noexcept
- static constexpr unsigned int **round_error** () noexcept
- static constexpr unsigned int signaling_NaN () noexcept
- static constexpr unsigned int **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int digits
- static constexpr int **digits**
- static constexpr int digits10
- static constexpr int **digits10**
- static constexpr float_denorm_style has_denorm
- static constexpr float_denorm_style **has_denorm**
- static constexpr bool has_denorm_loss
- static constexpr bool **has_denorm_loss**
- static constexpr bool has_infinity
- static constexpr bool **has_infinity**
- static constexpr bool has_quiet_NaN
- static constexpr bool **has_quiet_NaN**
- static constexpr bool has_signaling_NaN
- static constexpr bool **has_signaling_NaN**
- static constexpr bool is_bounded
- static constexpr bool **is_bounded**
- static constexpr bool is_exact
- static constexpr bool **is_exact**
- static constexpr bool is_iec559
- static constexpr bool **is_iec559**
- static constexpr bool is_integer
- static constexpr bool **is_integer**
- static constexpr bool is_modulo
- static constexpr bool **is_modulo**
- static constexpr bool is_signed
- static constexpr bool **is_signed**
- static constexpr bool is_specialized
- static constexpr bool **is_specialized**
- static constexpr int max_digits10

- static constexpr int **max_digits10**
- static constexpr int max_exponent
- static constexpr int **max_exponent**
- static constexpr int max_exponent10
- static constexpr int **max_exponent10**
- static constexpr int min_exponent
- static constexpr int **min_exponent**
- static constexpr int min_exponent10
- static constexpr int **min_exponent10**
- static constexpr int radix
- static constexpr int **radix**
- static constexpr float_round_style round_style
- static constexpr float_round_style **round_style**
- static constexpr bool tinyness_before
- static constexpr bool **tinyness_before**
- static constexpr bool traps
- static constexpr bool **traps**

5.724.1 Detailed Description

numeric_limits<unsigned int> specialization.

5.724.2 Member Function Documentation

denorm_min()

```
static constexpr unsigned int std::numeric_limits< unsigned int >::denorm_min ()  [inline], [static], [constexpr], [noexcept]
```

The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

epsilon()

```
static constexpr unsigned int std::numeric_limits< unsigned int >::epsilon ()  [inline], [static], [constexpr], [noexcept]
```

The *machine epsilon:* the difference between 1 and the least value greater than 1 that is representable.

infinity()

```
static constexpr unsigned int std::numeric_limits< unsigned int >::infinity ()  [inline], [static], [constexpr], [noexcept]
```

The representation of positive infinity, if `has_infinity`.

lowest()

```
static constexpr unsigned int std::numeric_limits< unsigned int >::lowest ()  [inline], [static], [constexpr], [noexcept]
```

A finite value x such that there is no other finite value y where y < x.

max()

```
static constexpr unsigned int std::numeric_limits< unsigned int >::max ()  [inline], [static], [constexpr], [noexcept]
```

The maximum finite value.

min()

static constexpr unsigned int std::numeric_limits< unsigned int >::min () [inline], [static], [constexpr], [noexcept]
The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

quiet_NaN()

static constexpr unsigned int std::numeric_limits< unsigned int >::quiet_NaN () [inline], [static], [constexpr], [noexcept]
The representation of a quiet Not a Number, if has_quiet_NaN.

round_error()

static constexpr unsigned int std::numeric_limits< unsigned int >::round_error () [inline], [static], [constexpr], [noexcept]
The maximum rounding error measurement (see LIA-1).

signaling_NaN()

static constexpr unsigned int std::numeric_limits< unsigned int >::signaling_NaN () [inline], [static], [constexpr], [noexcept]
The representation of a signaling Not a Number, if has_signaling_NaN.

5.724.3 Member Data Documentation

digits

int std::__numeric_limits_base::digits [static], [constexpr]
The number of radix digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of radix digits in the mantissa.

digits10

int std::__numeric_limits_base::digits10 [static], [constexpr]
The number of base 10 digits that can be represented without change.

has_denorm

float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
See std::float_denorm_style for more information.

has_denorm_loss

bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

has_infinity

bool std::__numeric_limits_base::has_infinity [static], [constexpr]
True if the type has a representation for positive infinity.

has_quiet_NaN

bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
True if the type has a representation for a quiet (non-signaling) Not a Number.

has_signaling_NaN

`bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]`
True if the type has a representation for a signaling Not a Number.

is_bounded

`bool std::__numeric_limits_base::is_bounded [static], [constexpr]`
True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

is_exact

`bool std::__numeric_limits_base::is_exact [static], [constexpr]`
True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

is_iec559

`bool std::__numeric_limits_base::is_iec559 [static], [constexpr]`
True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

is_integer

`bool std::__numeric_limits_base::is_integer [static], [constexpr]`
True if the type is integer.

is_modulo

`bool std::__numeric_limits_base::is_modulo [static], [constexpr]`
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or $*$ on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

is_signed

`bool std::__numeric_limits_base::is_signed [static], [constexpr]`
True if the type is signed.

is_specialized

`bool std::__numeric_limits_base::is_specialized [static], [constexpr]`
This will be true for all fundamental types (which have specializations), and false for everything else.

max_digits10

`int std::__numeric_limits_base::max_digits10 [static], [constexpr]`
The number of base 10 digits required to ensure that values which differ are always differentiated.

max_exponent

`int std::__numeric_limits_base::max_exponent [static], [constexpr]`
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

max_exponent10

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr]`
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

min_exponent

`int std::__numeric_limits_base::min_exponent [static], [constexpr]`
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

min_exponent10

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr]`
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

radix

`int std::__numeric_limits_base::radix [static], [constexpr]`
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

round_style

`float_round_style std::__numeric_limits_base::round_style [static], [constexpr]`
See std::float_round_style for more information. This is only meaningful for floating types; integer types will all be round_toward_zero.

tinyness_before

`bool std::__numeric_limits_base::tinyness_before [static], [constexpr]`
True if tininess is detected before rounding. (see IEC 559)

traps

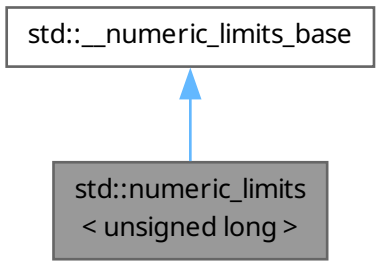
`bool std::__numeric_limits_base::traps [static], [constexpr]`
True if trapping is implemented for this type.
The documentation for this struct was generated from the following file:

- limits

5.725 std::numeric_limits< unsigned long > Struct Reference

`#include <limits>`

Inheritance diagram for std::numeric_limits< unsigned long >:



Static Public Member Functions

- static constexpr unsigned long denorm_min () noexcept
- static constexpr unsigned long **denorm_min** () noexcept
- static constexpr unsigned long epsilon () noexcept
- static constexpr unsigned long **epsilon** () noexcept
- static constexpr unsigned long infinity () noexcept
- static constexpr unsigned long **infinity** () noexcept
- static constexpr unsigned long lowest () noexcept
- static constexpr unsigned long **lowest** () noexcept
- static constexpr unsigned long max () noexcept
- static constexpr unsigned long **max** () noexcept
- static constexpr unsigned long min () noexcept
- static constexpr unsigned long **min** () noexcept
- static constexpr unsigned long quiet_NaN () noexcept
- static constexpr unsigned long **quiet_NaN** () noexcept
- static constexpr unsigned long round_error () noexcept
- static constexpr unsigned long **round_error** () noexcept
- static constexpr unsigned long signaling_NaN () noexcept
- static constexpr unsigned long **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int digits
- static constexpr int **digits**
- static constexpr int digits10
- static constexpr int **digits10**
- static constexpr float_denorm_style has_denorm
- static constexpr float_denorm_style **has_denorm**
- static constexpr bool has_denorm_loss
- static constexpr bool **has_denorm_loss**
- static constexpr bool has_infinity
- static constexpr bool **has_infinity**
- static constexpr bool has_quiet_NaN

- static constexpr bool **has_quiet_NaN**
- static constexpr bool has_signaling_NaN
- static constexpr bool **has_signaling_NaN**
- static constexpr bool is_bounded
- static constexpr bool **is_bounded**
- static constexpr bool is_exact
- static constexpr bool **is_exact**
- static constexpr bool is_iec559
- static constexpr bool **is_iec559**
- static constexpr bool is_integer
- static constexpr bool **is_integer**
- static constexpr bool is_modulo
- static constexpr bool **is_modulo**
- static constexpr bool is_signed
- static constexpr bool **is_signed**
- static constexpr bool is_specialized
- static constexpr bool **is_specialized**
- static constexpr int max_digits10
- static constexpr int **max_digits10**
- static constexpr int max_exponent
- static constexpr int **max_exponent**
- static constexpr int max_exponent10
- static constexpr int **max_exponent10**
- static constexpr int min_exponent
- static constexpr int **min_exponent**
- static constexpr int min_exponent10
- static constexpr int **min_exponent10**
- static constexpr int radix
- static constexpr int **radix**
- static constexpr float_round_style round_style
- static constexpr float_round_style **round_style**
- static constexpr bool tinyness_before
- static constexpr bool **tinyness_before**
- static constexpr bool traps
- static constexpr bool **traps**

5.725.1 Detailed Description

numeric_limits<unsigned long> specialization.

5.725.2 Member Function Documentation

denorm_min()

```
static constexpr unsigned long std::numeric_limits< unsigned long >::denorm_min ()  [inline],
[static], [constexpr], [noexcept]
```

The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

epsilon()

```
static constexpr unsigned long std::numeric_limits< unsigned long >::epsilon ()  [inline], [static],
[constexpr], [noexcept]
```

The *machine epsilon:* the difference between 1 and the least value greater than 1 that is representable.

infinity()

`static constexpr unsigned long std::numeric_limits< unsigned long >::infinity () [inline], [static], [constexpr], [noexcept]`
The representation of positive infinity, if `has_infinity`.

lowest()

`static constexpr unsigned long std::numeric_limits< unsigned long >::lowest () [inline], [static], [constexpr], [noexcept]`
A finite value x such that there is no other finite value y where y $<$ x.

max()

`static constexpr unsigned long std::numeric_limits< unsigned long >::max () [inline], [static], [constexpr], [noexcept]`
The maximum finite value.

min()

`static constexpr unsigned long std::numeric_limits< unsigned long >::min () [inline], [static], [constexpr], [noexcept]`
The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

quiet_NaN()

`static constexpr unsigned long std::numeric_limits< unsigned long >::quiet_NaN () [inline], [static], [constexpr], [noexcept]`
The representation of a quiet Not a Number, if `has_quiet_NaN`.

round_error()

`static constexpr unsigned long std::numeric_limits< unsigned long >::round_error () [inline], [static], [constexpr], [noexcept]`
The maximum rounding error measurement (see LIA-1).

signaling_NaN()

`static constexpr unsigned long std::numeric_limits< unsigned long >::signaling_NaN () [inline], [static], [constexpr], [noexcept]`
The representation of a signaling Not a Number, if `has_signaling_NaN`.

5.725.3 Member Data Documentation

digits

`int std::__numeric_limits_base::digits [static], [constexpr]`
The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

digits10

`int std::__numeric_limits_base::digits10 [static], [constexpr]`
The number of base 10 digits that can be represented without change.

has_denorm

float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
See std::float_denorm_style for more information.

has_denorm_loss

bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

has_infinity

bool std::__numeric_limits_base::has_infinity [static], [constexpr]
True if the type has a representation for positive infinity.

has_quiet_NaN

bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
True if the type has a representation for a quiet (non-signaling) Not a Number.

has_signaling_NaN

bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
True if the type has a representation for a signaling Not a Number.

is_bounded

bool std::__numeric_limits_base::is_bounded [static], [constexpr]
True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

is_exact

bool std::__numeric_limits_base::is_exact [static], [constexpr]
True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

is_iec559

bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

is_integer

bool std::__numeric_limits_base::is_integer [static], [constexpr]
True if the type is integer.

is_modulo

bool std::__numeric_limits_base::is_modulo [static], [constexpr]
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or $*$ on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

is_signed

`bool std::__numeric_limits_base::is_signed [static], [constexpr]`
True if the type is signed.

is_specialized

`bool std::__numeric_limits_base::is_specialized [static], [constexpr]`
This will be true for all fundamental types (which have specializations), and false for everything else.

max_digits10

`int std::__numeric_limits_base::max_digits10 [static], [constexpr]`
The number of base 10 digits required to ensure that values which differ are always differentiated.

max_exponent

`int std::__numeric_limits_base::max_exponent [static], [constexpr]`
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

max_exponent10

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr]`
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

min_exponent

`int std::__numeric_limits_base::min_exponent [static], [constexpr]`
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

min_exponent10

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr]`
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

radix

`int std::__numeric_limits_base::radix [static], [constexpr]`
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

round_style

`float_round_style std::__numeric_limits_base::round_style [static], [constexpr]`
See std::float_round_style for more information. This is only meaningful for floating types; integer types will all be round_toward_zero.

tinyness_before

`bool std::__numeric_limits_base::tinyness_before [static], [constexpr]`
True if tininess is detected before rounding. (see IEC 559)

traps

```
bool std::__numeric_limits_base::traps  [static], [constexpr]
```

True if trapping is implemented for this type.

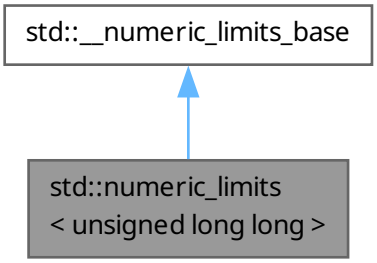
The documentation for this struct was generated from the following file:

- limits

5.726 std::numeric_limits< unsigned long long > Struct Reference

```
#include <limits>
```

Inheritance diagram for std::numeric_limits< unsigned long long >:



Static Public Member Functions

- static constexpr unsigned long long denorm_min () noexcept
- static constexpr unsigned long long **denorm_min** () noexcept
- static constexpr unsigned long long epsilon () noexcept
- static constexpr unsigned long long **epsilon** () noexcept
- static constexpr unsigned long long infinity () noexcept
- static constexpr unsigned long long **infinity** () noexcept
- static constexpr unsigned long long lowest () noexcept
- static constexpr unsigned long long **lowest** () noexcept
- static constexpr unsigned long long max () noexcept
- static constexpr unsigned long long **max** () noexcept
- static constexpr unsigned long long min () noexcept
- static constexpr unsigned long long **min** () noexcept
- static constexpr unsigned long long quiet_NaN () noexcept
- static constexpr unsigned long long **quiet_NaN** () noexcept
- static constexpr unsigned long long round_error () noexcept
- static constexpr unsigned long long **round_error** () noexcept
- static constexpr unsigned long long signaling_NaN () noexcept
- static constexpr unsigned long long **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int digits
- static constexpr int **digits**
- static constexpr int digits10
- static constexpr int **digits10**
- static constexpr float_denorm_style has_denorm
- static constexpr float_denorm_style **has_denorm**
- static constexpr bool has_denorm_loss
- static constexpr bool **has_denorm_loss**
- static constexpr bool has_infinity
- static constexpr bool **has_infinity**
- static constexpr bool has_quiet_NaN
- static constexpr bool **has_quiet_NaN**
- static constexpr bool has_signaling_NaN
- static constexpr bool **has_signaling_NaN**
- static constexpr bool is_bounded
- static constexpr bool **is_bounded**
- static constexpr bool is_exact
- static constexpr bool **is_exact**
- static constexpr bool is_iec559
- static constexpr bool **is_iec559**
- static constexpr bool is_integer
- static constexpr bool **is_integer**
- static constexpr bool is_modulo
- static constexpr bool **is_modulo**
- static constexpr bool is_signed
- static constexpr bool **is_signed**
- static constexpr bool is_specialized
- static constexpr bool **is_specialized**
- static constexpr int max_digits10
- static constexpr int **max_digits10**
- static constexpr int max_exponent
- static constexpr int **max_exponent**
- static constexpr int max_exponent10
- static constexpr int **max_exponent10**
- static constexpr int min_exponent
- static constexpr int **min_exponent**
- static constexpr int min_exponent10
- static constexpr int **min_exponent10**
- static constexpr int radix
- static constexpr int **radix**
- static constexpr float_round_style round_style
- static constexpr float_round_style **round_style**
- static constexpr bool tinyness_before
- static constexpr bool **tinyness_before**
- static constexpr bool traps
- static constexpr bool **traps**

5.726.1 Detailed Description

numeric_limits<unsigned long long> specialization.

5.726.2 Member Function Documentation

denorm_min()

`static constexpr unsigned long long std::numeric_limits< unsigned long long >::denorm_min () [inline], [static], [constexpr], [noexcept]`
The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

epsilon()

`static constexpr unsigned long long std::numeric_limits< unsigned long long >::epsilon () [inline], [static], [constexpr], [noexcept]`
The *machine epsilon:* the difference between 1 and the least value greater than 1 that is representable.

infinity()

`static constexpr unsigned long long std::numeric_limits< unsigned long long >::infinity () [inline], [static], [constexpr], [noexcept]`
The representation of positive infinity, if `has_infinity`.

lowest()

`static constexpr unsigned long long std::numeric_limits< unsigned long long >::lowest () [inline], [static], [constexpr], [noexcept]`
A finite value x such that there is no other finite value y where y < x.

max()

`static constexpr unsigned long long std::numeric_limits< unsigned long long >::max () [inline], [static], [constexpr], [noexcept]`
The maximum finite value.

min()

`static constexpr unsigned long long std::numeric_limits< unsigned long long >::min () [inline], [static], [constexpr], [noexcept]`
The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

quiet_NaN()

`static constexpr unsigned long long std::numeric_limits< unsigned long long >::quiet_NaN () [inline], [static], [constexpr], [noexcept]`
The representation of a quiet Not a Number, if `has_quiet_NaN`.

round_error()

`static constexpr unsigned long long std::numeric_limits< unsigned long long >::round_error () [inline], [static], [constexpr], [noexcept]`
The maximum rounding error measurement (see LIA-1).

signaling_NaN()

`static constexpr unsigned long long std::numeric_limits< unsigned long long >::signaling_NaN () [inline], [static], [constexpr], [noexcept]`
The representation of a signaling Not a Number, if `has_signaling_NaN`.

5.726.3 Member Data Documentation

digits

`int std::__numeric_limits_base::digits [static], [constexpr]`
The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

digits10

`int std::__numeric_limits_base::digits10 [static], [constexpr]`
The number of base 10 digits that can be represented without change.

has_denorm

`float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]`
See std::float_denorm_style for more information.

has_denorm_loss

`bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]`
True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

has_infinity

`bool std::__numeric_limits_base::has_infinity [static], [constexpr]`
True if the type has a representation for positive infinity.

has_quiet_NaN

`bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]`
True if the type has a representation for a quiet (non-signaling) Not a Number.

has_signaling_NaN

`bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]`
True if the type has a representation for a signaling Not a Number.

is_bounded

`bool std::__numeric_limits_base::is_bounded [static], [constexpr]`
True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

is_exact

`bool std::__numeric_limits_base::is_exact [static], [constexpr]`
True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

is_iec559

`bool std::__numeric_limits_base::is_iec559 [static], [constexpr]`
True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

is_integer

bool std::__numeric_limits_base::is_integer [static], [constexpr]
True if the type is integer.

is_modulo

bool std::__numeric_limits_base::is_modulo [static], [constexpr]
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or * on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

is_signed

bool std::__numeric_limits_base::is_signed [static], [constexpr]
True if the type is signed.

is_specialized

bool std::__numeric_limits_base::is_specialized [static], [constexpr]
This will be true for all fundamental types (which have specializations), and false for everything else.

max_digits10

int std::__numeric_limits_base::max_digits10 [static], [constexpr]
The number of base 10 digits required to ensure that values which differ are always differentiated.

max_exponent

int std::__numeric_limits_base::max_exponent [static], [constexpr]
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

max_exponent10

int std::__numeric_limits_base::max_exponent10 [static], [constexpr]
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

min_exponent

int std::__numeric_limits_base::min_exponent [static], [constexpr]
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

min_exponent10

int std::__numeric_limits_base::min_exponent10 [static], [constexpr]
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

radix

int std::__numeric_limits_base::radix [static], [constexpr]
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

round_style

`float_round_style std::__numeric_limits_base::round_style [static], [constexpr]`
See std::float_round_style for more information. This is only meaningful for floating types; integer types will all be round_toward_zero.

tinyness_before

`bool std::__numeric_limits_base::tinyness_before [static], [constexpr]`
True if tininess is detected before rounding. (see IEC 559)

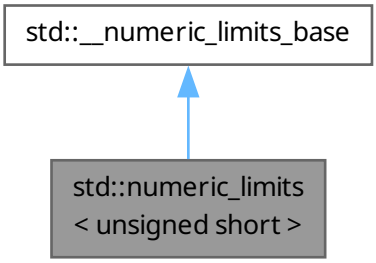
traps

`bool std::__numeric_limits_base::traps [static], [constexpr]`
True if trapping is implemented for this type.
The documentation for this struct was generated from the following file:

- limits

5.727 std::numeric_limits< unsigned short > Struct Reference

`#include <limits>`
Inheritance diagram for std::numeric_limits< unsigned short >:



Static Public Member Functions

- static constexpr unsigned short denorm_min () noexcept
- static constexpr unsigned short **denorm_min** () noexcept
- static constexpr unsigned short epsilon () noexcept
- static constexpr unsigned short **epsilon** () noexcept
- static constexpr unsigned short infinity () noexcept
- static constexpr unsigned short **infinity** () noexcept
- static constexpr unsigned short lowest () noexcept
- static constexpr unsigned short **lowest** () noexcept
- static constexpr unsigned short max () noexcept
- static constexpr unsigned short **max** () noexcept
- static constexpr unsigned short min () noexcept
- static constexpr unsigned short **min** () noexcept

- static constexpr unsigned short quiet_NaN () noexcept
- static constexpr unsigned short **quiet_NaN** () noexcept
- static constexpr unsigned short round_error () noexcept
- static constexpr unsigned short **round_error** () noexcept
- static constexpr unsigned short signaling_NaN () noexcept
- static constexpr unsigned short **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int digits
- static constexpr int **digits**
- static constexpr int digits10
- static constexpr int **digits10**
- static constexpr float_denorm_style has_denorm
- static constexpr float_denorm_style **has_denorm**
- static constexpr bool has_denorm_loss
- static constexpr bool **has_denorm_loss**
- static constexpr bool has_infinity
- static constexpr bool **has_infinity**
- static constexpr bool has_quiet_NaN
- static constexpr bool **has_quiet_NaN**
- static constexpr bool has_signaling_NaN
- static constexpr bool **has_signaling_NaN**
- static constexpr bool is_bounded
- static constexpr bool **is_bounded**
- static constexpr bool is_exact
- static constexpr bool **is_exact**
- static constexpr bool is_iec559
- static constexpr bool **is_iec559**
- static constexpr bool is_integer
- static constexpr bool **is_integer**
- static constexpr bool is_modulo
- static constexpr bool **is_modulo**
- static constexpr bool is_signed
- static constexpr bool **is_signed**
- static constexpr bool is_specialized
- static constexpr bool **is_specialized**
- static constexpr int max_digits10
- static constexpr int **max_digits10**
- static constexpr int max_exponent
- static constexpr int **max_exponent**
- static constexpr int max_exponent10
- static constexpr int **max_exponent10**
- static constexpr int min_exponent
- static constexpr int **min_exponent**
- static constexpr int min_exponent10
- static constexpr int **min_exponent10**
- static constexpr int radix
- static constexpr int **radix**
- static constexpr float_round_style round_style
- static constexpr float_round_style **round_style**

- static constexpr bool tinyness_before
- static constexpr bool **tinyness_before**
- static constexpr bool traps
- static constexpr bool **traps**

5.727.1 Detailed Description

numeric_limits<unsigned short> specialization.

5.727.2 Member Function Documentation

denorm_min()

`static constexpr unsigned short std::numeric_limits< unsigned short >::denorm_min () [inline], [static], [constexpr], [noexcept]`
The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

epsilon()

`static constexpr unsigned short std::numeric_limits< unsigned short >::epsilon () [inline], [static], [constexpr], [noexcept]`
The *machine epsilon:* the difference between 1 and the least value greater than 1 that is representable.

infinity()

`static constexpr unsigned short std::numeric_limits< unsigned short >::infinity () [inline], [static], [constexpr], [noexcept]`
The representation of positive infinity, if `has_infinity`.

lowest()

`static constexpr unsigned short std::numeric_limits< unsigned short >::lowest () [inline], [static], [constexpr], [noexcept]`
A finite value x such that there is no other finite value y where y < x.

max()

`static constexpr unsigned short std::numeric_limits< unsigned short >::max () [inline], [static], [constexpr], [noexcept]`
The maximum finite value.

min()

`static constexpr unsigned short std::numeric_limits< unsigned short >::min () [inline], [static], [constexpr], [noexcept]`
The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

quiet_NaN()

`static constexpr unsigned short std::numeric_limits< unsigned short >::quiet_NaN () [inline], [static], [constexpr], [noexcept]`
The representation of a quiet Not a Number, if `has_quiet_NaN`.

round_error()

static constexpr unsigned short std::numeric_limits< unsigned short >::round_error () [inline], [static], [constexpr], [noexcept]
The maximum rounding error measurement (see LIA-1).

signaling_NaN()

static constexpr unsigned short std::numeric_limits< unsigned short >::signaling_NaN () [inline], [static], [constexpr], [noexcept]
The representation of a signaling Not a Number, if has_signaling_NaN.

5.727.3 Member Data Documentation

digits

int std::__numeric_limits_base::digits [static], [constexpr]
The number of radix digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of radix digits in the mantissa.

digits10

int std::__numeric_limits_base::digits10 [static], [constexpr]
The number of base 10 digits that can be represented without change.

has_denorm

float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
See std::float_denorm_style for more information.

has_denorm_loss

bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

has_infinity

bool std::__numeric_limits_base::has_infinity [static], [constexpr]
True if the type has a representation for positive infinity.

has_quiet_NaN

bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
True if the type has a representation for a quiet (non-signaling) Not a Number.

has_signaling_NaN

bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
True if the type has a representation for a signaling Not a Number.

is_bounded

bool std::__numeric_limits_base::is_bounded [static], [constexpr]
True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

is_exact

`bool std::__numeric_limits_base::is_exact [static], [constexpr]`
True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

is_iec559

`bool std::__numeric_limits_base::is_iec559 [static], [constexpr]`
True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

is_integer

`bool std::__numeric_limits_base::is_integer [static], [constexpr]`
True if the type is integer.

is_modulo

`bool std::__numeric_limits_base::is_modulo [static], [constexpr]`
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or $*$ on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

is_signed

`bool std::__numeric_limits_base::is_signed [static], [constexpr]`
True if the type is signed.

is_specialized

`bool std::__numeric_limits_base::is_specialized [static], [constexpr]`
This will be true for all fundamental types (which have specializations), and false for everything else.

max_digits10

`int std::__numeric_limits_base::max_digits10 [static], [constexpr]`
The number of base 10 digits required to ensure that values which differ are always differentiated.

max_exponent

`int std::__numeric_limits_base::max_exponent [static], [constexpr]`
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

max_exponent10

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr]`
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

min_exponent

```
int std::__numeric_limits_base::min_exponent  [static], [constexpr]
```
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

min_exponent10

```
int std::__numeric_limits_base::min_exponent10  [static], [constexpr]
```
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

radix

```
int std::__numeric_limits_base::radix  [static], [constexpr]
```
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

round_style

```
float_round_style std::__numeric_limits_base::round_style  [static], [constexpr]
```
See std::float_round_style for more information. This is only meaningful for floating types; integer types will all be round_toward_zero.

tinyness_before

```
bool std::__numeric_limits_base::tinyness_before  [static], [constexpr]
```
True if tininess is detected before rounding. (see IEC 559)

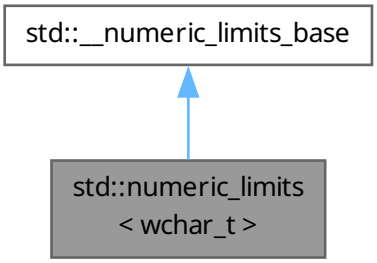
traps

```
bool std::__numeric_limits_base::traps  [static], [constexpr]
```
True if trapping is implemented for this type.
The documentation for this struct was generated from the following file:

- limits

5.728 std::numeric_limits< wchar_t > Struct Reference

`#include <limits>`
Inheritance diagram for std::numeric_limits< wchar_t >:



Static Public Member Functions

- static constexpr wchar_t denorm_min () noexcept
- static constexpr wchar_t **denorm_min** () noexcept
- static constexpr wchar_t epsilon () noexcept
- static constexpr wchar_t **epsilon** () noexcept
- static constexpr wchar_t infinity () noexcept
- static constexpr wchar_t **infinity** () noexcept
- static constexpr wchar_t lowest () noexcept
- static constexpr wchar_t **lowest** () noexcept
- static constexpr wchar_t max () noexcept
- static constexpr wchar_t **max** () noexcept
- static constexpr wchar_t min () noexcept
- static constexpr wchar_t **min** () noexcept
- static constexpr wchar_t quiet_NaN () noexcept
- static constexpr wchar_t **quiet_NaN** () noexcept
- static constexpr wchar_t round_error () noexcept
- static constexpr wchar_t **round_error** () noexcept
- static constexpr wchar_t signaling_NaN () noexcept
- static constexpr wchar_t **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int digits
- static constexpr int **digits**
- static constexpr int digits10
- static constexpr int **digits10**
- static constexpr float_denorm_style has_denorm
- static constexpr float_denorm_style **has_denorm**
- static constexpr bool has_denorm_loss
- static constexpr bool **has_denorm_loss**
- static constexpr bool has_infinity
- static constexpr bool **has_infinity**
- static constexpr bool has_quiet_NaN
- static constexpr bool **has_quiet_NaN**
- static constexpr bool has_signaling_NaN
- static constexpr bool **has_signaling_NaN**
- static constexpr bool is_bounded
- static constexpr bool **is_bounded**
- static constexpr bool is_exact
- static constexpr bool **is_exact**
- static constexpr bool is_iec559
- static constexpr bool **is_iec559**
- static constexpr bool is_integer
- static constexpr bool **is_integer**
- static constexpr bool is_modulo
- static constexpr bool **is_modulo**
- static constexpr bool is_signed
- static constexpr bool **is_signed**
- static constexpr bool is_specialized
- static constexpr bool **is_specialized**
- static constexpr int max_digits10

- static constexpr int **max_digits10**
- static constexpr int max_exponent
- static constexpr int **max_exponent**
- static constexpr int max_exponent10
- static constexpr int **max_exponent10**
- static constexpr int min_exponent
- static constexpr int **min_exponent**
- static constexpr int min_exponent10
- static constexpr int **min_exponent10**
- static constexpr int radix
- static constexpr int **radix**
- static constexpr float_round_style round_style
- static constexpr float_round_style **round_style**
- static constexpr bool tinyness_before
- static constexpr bool **tinyness_before**
- static constexpr bool traps
- static constexpr bool **traps**

5.728.1 Detailed Description

numeric_limits<wchar_t> specialization.

5.728.2 Member Function Documentation

denorm_min()

```
static constexpr wchar_t std::numeric_limits< wchar_t >::denorm_min ()  [inline], [static], [constexpr], [noexcept]
```

The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

epsilon()

```
static constexpr wchar_t std::numeric_limits< wchar_t >::epsilon ()  [inline], [static], [constexpr], [noexcept]
```

The *machine epsilon:* the difference between 1 and the least value greater than 1 that is representable.

infinity()

```
static constexpr wchar_t std::numeric_limits< wchar_t >::infinity ()  [inline], [static], [constexpr], [noexcept]
```

The representation of positive infinity, if `has_infinity`.

lowest()

```
static constexpr wchar_t std::numeric_limits< wchar_t >::lowest ()  [inline], [static], [constexpr], [noexcept]
```

A finite value x such that there is no other finite value y where y < x.

max()

```
static constexpr wchar_t std::numeric_limits< wchar_t >::max ()  [inline], [static], [constexpr], [noexcept]
```

The maximum finite value.

min()

```
static constexpr wchar_t std::numeric_limits< wchar_t >::min ()  [inline], [static], [constexpr], [noexcept]
```
The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

quiet_NaN()

```
static constexpr wchar_t std::numeric_limits< wchar_t >::quiet_NaN ()  [inline], [static], [constexpr], [noexcept]
```
The representation of a quiet Not a Number, if `has_quiet_NaN`.

round_error()

```
static constexpr wchar_t std::numeric_limits< wchar_t >::round_error ()  [inline], [static], [constexpr], [noexcept]
```
The maximum rounding error measurement (see LIA-1).

signaling_NaN()

```
static constexpr wchar_t std::numeric_limits< wchar_t >::signaling_NaN ()  [inline], [static], [constexpr], [noexcept]
```
The representation of a signaling Not a Number, if `has_signaling_NaN`.

5.728.3 Member Data Documentation

digits

```
int std::__numeric_limits_base::digits  [static], [constexpr]
```
The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.

digits10

```
int std::__numeric_limits_base::digits10  [static], [constexpr]
```
The number of base 10 digits that can be represented without change.

has_denorm

```
float_denorm_style std::__numeric_limits_base::has_denorm  [static], [constexpr]
```
See std::float_denorm_style for more information.

has_denorm_loss

```
bool std::__numeric_limits_base::has_denorm_loss  [static], [constexpr]
```
True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

has_infinity

```
bool std::__numeric_limits_base::has_infinity  [static], [constexpr]
```
True if the type has a representation for positive infinity.

has_quiet_NaN

```
bool std::__numeric_limits_base::has_quiet_NaN  [static], [constexpr]
```
True if the type has a representation for a quiet (non-signaling) Not a Number.

has_signaling_NaN

bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
True if the type has a representation for a signaling Not a Number.

is_bounded

bool std::__numeric_limits_base::is_bounded [static], [constexpr]
True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

is_exact

bool std::__numeric_limits_base::is_exact [static], [constexpr]
True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

is_iec559

bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

is_integer

bool std::__numeric_limits_base::is_integer [static], [constexpr]
True if the type is integer.

is_modulo

bool std::__numeric_limits_base::is_modulo [static], [constexpr]
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or $*$ on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

is_signed

bool std::__numeric_limits_base::is_signed [static], [constexpr]
True if the type is signed.

is_specialized

bool std::__numeric_limits_base::is_specialized [static], [constexpr]
This will be true for all fundamental types (which have specializations), and false for everything else.

max_digits10

int std::__numeric_limits_base::max_digits10 [static], [constexpr]
The number of base 10 digits required to ensure that values which differ are always differentiated.

max_exponent

int std::__numeric_limits_base::max_exponent [static], [constexpr]
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

max_exponent10

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr]`
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

min_exponent

`int std::__numeric_limits_base::min_exponent [static], [constexpr]`
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

min_exponent10

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr]`
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

radix

`int std::__numeric_limits_base::radix [static], [constexpr]`
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

round_style

`float_round_style std::__numeric_limits_base::round_style [static], [constexpr]`
See std::float_round_style for more information. This is only meaningful for floating types; integer types will all be round_toward_zero.

tinyness_before

`bool std::__numeric_limits_base::tinyness_before [static], [constexpr]`
True if tininess is detected before rounding. (see IEC 559)

traps

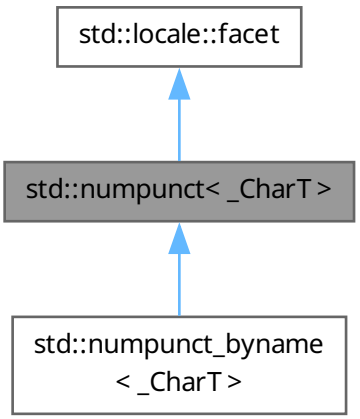
`bool std::__numeric_limits_base::traps [static], [constexpr]`
True if trapping is implemented for this type.
The documentation for this struct was generated from the following file:

- limits

5.729 std::numpunct< _CharT > Class Template Reference

`#include <locale_facets.h>`

Inheritance diagram for std::numpunct< _CharT >:



Public Types

- typedef __numpunct_cache< _CharT > **__cache_type**

- typedef _CharT char_type
- typedef basic_string< _CharT > string_type

Public Member Functions

- numpunct (__c_locale __cloc, size_t __refs=0)
- numpunct (__cache_type *__cache, size_t __refs=0)
- numpunct (size_t __refs=0)
- char_type decimal_point () const
- string_type falsename () const
- string grouping () const
- char_type thousands_sep () const
- string_type truename () const

Static Public Attributes

- static locale::id id

Protected Member Functions

- virtual ~numpunct ()
- void **_M_initialize_numpunct** (__c_locale __cloc)
- void **_M_initialize_numpunct** (__c_locale __cloc)
- void **_M_initialize_numpunct** (__c_locale __cloc=0)
- virtual char_type do_decimal_point () const
- virtual string_type do_falsename () const

- virtual string do_grouping () const
- virtual char_type do_thousands_sep () const
- virtual string_type do_truename () const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __cache_type * **_M_data**

5.729.1 Detailed Description

template<typename _CharT>
class std::numpunct< _CharT >

Primary class template numpunct.
This facet stores several pieces of information related to printing and scanning numbers, such as the decimal point character. It takes a template parameter specifying the char type. The numpunct facet is used by streams for many I/O operations involving numbers.
The numpunct template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from a numpunct facet.

5.729.2 Member Typedef Documentation

char_type

```
template<typename _CharT>
typedef _CharT std::numpunct< _CharT >::char_type
```

Public typedefs.

string_type

```
template<typename _CharT>
typedef basic_string<_CharT> std::numpunct< _CharT >::string_type
```

Public typedefs.

5.729.3 Constructor & Destructor Documentation

numpunct() [1/3]

```
template<typename _CharT>
std::numpunct< _CharT >::numpunct (
            size_t __refs = 0)  [inline], [explicit]
```

Numpunct constructor.

Parameters

__refs	Refcount to pass to the base class.

References std::locale::facet::facet().

numpunct() [2/3]

```
template<typename _CharT>
std::numpunct< _CharT >::numpunct (
            __cache_type * __cache,
            size_t __refs = 0)  [inline], [explicit]
```

Internal constructor. Not for general use.
This is a constructor for use by the library itself to set up the predefined locale facets.

Parameters

__cache	__numpunct_cache object.
__refs	Refcount to pass to the base class.

References std::locale::facet::facet().

numpunct() [3/3]

```
template<typename _CharT>
std::numpunct< _CharT >::numpunct (
            __c_locale __cloc,
            size_t __refs = 0)  [inline], [explicit]
```

Internal constructor. Not for general use.
This is a constructor for use by the library itself to set up new locales.

Parameters

__cloc	The C locale.
__refs	Refcount to pass to the base class.

References std::locale::facet::facet().

~numpunct()

```
template<typename _CharT>
virtual std::numpunct< _CharT >::~numpunct ()  [protected], [virtual]
```

Destructor.

5.729.4 Member Function Documentation

decimal_point()

```
template<typename _CharT>
char_type std::numpunct< _CharT >::decimal_point () const  [inline]
```

Return decimal point character.
This function returns a char_type to use as a decimal point. It does so by returning returning numpunct<char_type>↩ ::do_decimal_point().

Returns

char_type representing a decimal point.

References do_decimal_point().

do_decimal_point()

```
template<typename _CharT>
virtual char_type std::numpunct< _CharT >::do_decimal_point () const  [inline], [protected],
[virtual]
```

Return decimal point character.
Returns a char_type to use as a decimal point. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a decimal point.

Referenced by decimal_point().

do_falsename()

```
template<typename _CharT>
virtual string_type std::numpunct< _CharT >::do_falsename () const  [inline], [protected], [virtual]
```

Return string representation of bool false.
Returns a string_type containing the text representation for false bool variables. This function is a hook for derived classes to change the value returned.

Returns

string_type representing printed form of false.

Referenced by falsename().

do_grouping()

```
template<typename _CharT>
virtual string std::numpunct< _CharT >::do_grouping () const  [inline], [protected], [virtual]
```

Return grouping specification.
Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

See also

grouping() for details.

Returns

String representing grouping specification.

Referenced by grouping().

do_thousands_sep()

```
template<typename _CharT>
virtual char_type std::numpunct< _CharT >::do_thousands_sep () const  [inline], [protected],
[virtual]
```

Return thousands separator character.
Returns a char_type to use as a thousands separator. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a thousands separator.

Referenced by thousands_sep().

do_truename()

```
template<typename _CharT>
virtual string_type std::numpunct< _CharT >::do_truename () const  [inline], [protected], [virtual]
```

Return string representation of bool true.

Returns a string_type containing the text representation for true bool variables. This function is a hook for derived classes to change the value returned.

Returns

string_type representing printed form of true.

Referenced by truename().

falsename()

```
template<typename _CharT>
string_type std::numpunct< _CharT >::falsename () const  [inline]
```

Return string representation of bool false.

This function returns a string_type containing the text representation for false bool variables. It does so by calling numpunct<char_type>::do_falsename().

Returns

string_type representing printed form of false.

References do_falsename().

grouping()

```
template<typename _CharT>
string std::numpunct< _CharT >::grouping () const  [inline]
```

Return grouping specification.

This function returns a string representing groupings for the integer part of a number. Groupings indicate where thousands separators should be inserted in the integer part of a number.

Each char in the return string is interpret as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the grouping() returns "\003\002" and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was "32", this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling numpunct<char_type>::do_grouping().

Returns

string representing grouping specification.

References do_grouping().

thousands_sep()

```
template<typename _CharT>
char_type std::numpunct< _CharT >::thousands_sep () const  [inline]
```

Return thousands separator character.

This function returns a char_type to use as a thousands separator. It does so by returning returning numpunct<char↩ _type>::do_thousands_sep().

Returns

char_type representing a thousands separator.

References do_thousands_sep().

truename()

```
template<typename _CharT>
string_type std::numpunct< _CharT >::truename () const  [inline]
```

Return string representation of bool true.

This function returns a string_type containing the text representation for true bool variables. It does so by calling numpunct<char_type>::do_truename().

Returns

string_type representing printed form of true.

References do_truename().

5.729.5 Member Data Documentation

id

```
template<typename _CharT>
locale::id std::numpunct< _CharT >::id  [static]
```

Numpunct facet id.

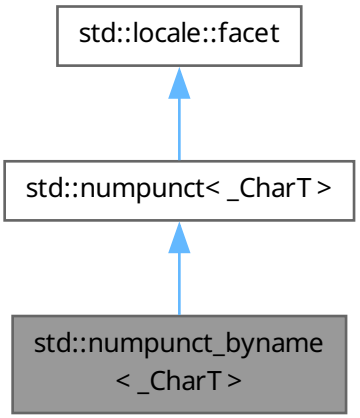
The documentation for this class was generated from the following file:

- locale_facets.h

5.730 std::numpunct_byname< _CharT > Class Template Reference

```
#include <locale_facets.h>
```

Inheritance diagram for std::numpunct_byname< _CharT >:



Public Types

- typedef __numpunct_cache< _CharT > **__cache_type**
- typedef _CharT **char_type**
- typedef basic_string< _CharT > **string_type**

Public Member Functions

- **numpunct_byname** (const char *__s, size_t __refs=0)
- **numpunct_byname** (const string &__s, size_t __refs=0)
- char_type decimal_point () const
- string_type falsename () const
- string grouping () const
- char_type thousands_sep () const
- string_type truename () const

Static Public Attributes

- static locale::id id

Protected Member Functions

- void **_M_initialize_numpunct** (__c_locale __cloc)
- void **_M_initialize_numpunct** (__c_locale __cloc)
- void **_M_initialize_numpunct** (__c_locale __cloc=0)
- virtual char_type do_decimal_point () const
- virtual string_type do_falsename () const
- virtual string do_grouping () const
- virtual char_type do_thousands_sep () const
- virtual string_type do_truename () const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __cache_type * **_M_data**

5.730.1 Detailed Description

template<typename _CharT>
class std::numpunct_byname< _CharT >

class numpunct_byname [22.2.3.2].

5.730.2 Member Function Documentation

decimal_point()

```
template<typename _CharT>
char_type std::numpunct< _CharT >::decimal_point () const  [inline], [inherited]
```

Return decimal point character.
This function returns a char_type to use as a decimal point. It does so by returning returning numpunct<char_type>↩ ::do_decimal_point().

Returns

char_type representing a decimal point.

References do_decimal_point().

do_decimal_point()

```
template<typename _CharT>
virtual char_type std::numpunct< _CharT >::do_decimal_point () const  [inline], [protected],
[virtual], [inherited]
```

Return decimal point character.
Returns a char_type to use as a decimal point. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a decimal point.

Referenced by decimal_point().

do_falsename()

```
template<typename _CharT>
virtual string_type std::numpunct< _CharT >::do_falsename () const  [inline], [protected], [virtual],
[inherited]
```

Return string representation of bool false.
Returns a string_type containing the text representation for false bool variables. This function is a hook for derived classes to change the value returned.

Returns

string_type representing printed form of false.

Referenced by falsename().

do_grouping()

```
template<typename _CharT>
virtual string std::numpunct< _CharT >::do_grouping () const  [inline], [protected], [virtual],
[inherited]
```

Return grouping specification.
Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

See also

grouping() for details.

Returns

String representing grouping specification.

Referenced by grouping().

do_thousands_sep()

```
template<typename _CharT>
virtual char_type std::numpunct< _CharT >::do_thousands_sep () const  [inline], [protected],
[virtual], [inherited]
```

Return thousands separator character.
Returns a char_type to use as a thousands separator. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a thousands separator.

Referenced by thousands_sep().

do_truename()

```
template<typename _CharT>
virtual string_type std::numpunct< _CharT >::do_truename () const  [inline], [protected], [virtual],
[inherited]
```

Return string representation of bool true.

Returns a string_type containing the text representation for true bool variables. This function is a hook for derived classes to change the value returned.

Returns

string_type representing printed form of true.

Referenced by truename().

falsename()

```
template<typename _CharT>
string_type std::numpunct< _CharT >::falsename () const  [inline], [inherited]
```

Return string representation of bool false.

This function returns a string_type containing the text representation for false bool variables. It does so by calling numpunct<char_type>::do_falsename().

Returns

string_type representing printed form of false.

References do_falsename().

grouping()

```
template<typename _CharT>
string std::numpunct< _CharT >::grouping () const  [inline], [inherited]
```

Return grouping specification.

This function returns a string representing groupings for the integer part of a number. Groupings indicate where thousands separators should be inserted in the integer part of a number.

Each char in the return string is interpret as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the grouping() returns "\003\002" and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was "32", this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling numpunct<char_type>::do_grouping().

Returns

string representing grouping specification.

References do_grouping().

thousands_sep()

```
template<typename _CharT>
char_type std::numpunct< _CharT >::thousands_sep () const  [inline], [inherited]
```

Return thousands separator character.
This function returns a char_type to use as a thousands separator. It does so by returning returning numpunct<char↩ _type>::do_thousands_sep().

Returns

char_type representing a thousands separator.

References do_thousands_sep().

truename()

```
template<typename _CharT>
string_type std::numpunct< _CharT >::truename () const  [inline], [inherited]
```

Return string representation of bool true.
This function returns a string_type containing the text representation for true bool variables. It does so by calling numpunct<char_type>::do_truename().

Returns

string_type representing printed form of true.

References do_truename().

5.730.3 Member Data Documentation

id

```
template<typename _CharT>
locale::id std::numpunct< _CharT >::id  [static], [inherited]
```

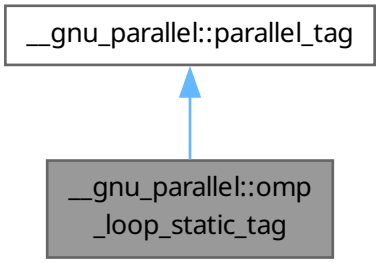
Numpunct facet id.
The documentation for this class was generated from the following file:

- locale_facets.h

5.731 __gnu_parallel::omp_loop_static_tag Struct Reference

```
#include <tags.h>
```

Inheritance diagram for __gnu_parallel::omp_loop_static_tag:



Public Member Functions

- _ThreadIndex __get_num_threads ()
- void set_num_threads (_ThreadIndex __num_threads)

5.731.1 Detailed Description

Recommends parallel execution using OpenMP static load-balancing at compile time.

5.731.2 Member Function Documentation

__get_num_threads()

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ()  [inline], [inherited]
```

Find out desired number of threads.

Returns

Desired number of threads.

Referenced by __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), and __gnu_parallel::__parallel_sort(

set_num_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
            _ThreadIndex __num_threads)  [inline], [inherited]
```

Set the desired number of threads.

Parameters

__num_threads	Desired number of threads.

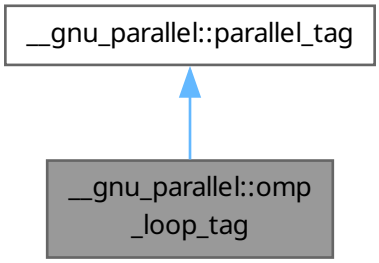
The documentation for this struct was generated from the following file:

- tags.h

5.732 __gnu_parallel::omp_loop_tag Struct Reference

```
#include <tags.h>
```

Inheritance diagram for __gnu_parallel::omp_loop_tag:



Public Member Functions

- _ThreadIndex __get_num_threads ()
- void set_num_threads (_ThreadIndex __num_threads)

5.732.1 Detailed Description

Recommends parallel execution using OpenMP dynamic load-balancing at compile time.

5.732.2 Member Function Documentation

__get_num_threads()

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ()  [inline], [inherited]
```

Find out desired number of threads.

Returns

Desired number of threads.

Referenced by __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), and __gnu_parallel::__parallel_sort(

set_num_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
            _ThreadIndex __num_threads)  [inline], [inherited]
```

Set the desired number of threads.

Parameters

__num_threads	Desired number of threads.

The documentation for this struct was generated from the following file:

- tags.h

5.733 std::once_flag Struct Reference

```
#include <mutex>
```

Public Member Functions

- once_flag (const once_flag &)=delete
- once_flag & operator= (const once_flag &)=delete

Friends

- template<typename _Callable, typename... _Args>
  void call_once (once_flag &__once, _Callable &&__f, _Args &&... __args)

5.733.1 Detailed Description

Flag type used by std::call_once.

5.733.2 Constructor & Destructor Documentation

once_flag()

```
std::once_flag::once_flag (
            const once_flag & )  [delete]
```

Deleted copy constructor.

5.733.3 Member Function Documentation

operator=()

```
once_flag & std::once_flag::operator= (
            const once_flag & )  [delete]
```

Deleted assignment operator.

5.733.4 Friends And Related Symbol Documentation

call_once

```
template<typename _Callable, typename...  _Args>
void call_once (
            once_flag & __once,
            _Callable && __f,
            _Args &&...  __args)  [friend]
```

Invoke a callable and synchronize with other calls using the same flag.
The documentation for this struct was generated from the following file:

- mutex

5.734 std::experimental::fundamentals_v1::optional< _Tp > Class Template Reference

`#include <optional>`

Public Types

- using **value_type**

Public Member Functions

- template<typename _Up = _Tp, enable_if_t< __and_< __not_< is_same< optional< _Tp >, decay_t< _Up > > >, is_constructible< _Tp, _Up && >, is_convertible< _Up &&, _Tp > >::value, bool > = true>
  constexpr **optional** (_Up &&__t)
- template<typename _Up = _Tp, enable_if_t< __and_< __not_< is_same< optional< _Tp >, decay_t< _Up > > >, is_constructible< _Tp, _Up && >, __not_< is_convertible< _Up &&, _Tp > > >::value, bool > = false>
  constexpr **optional** (_Up &&__t)
- template<typename _Up, enable_if_t< __and_< __not_< is_same< _Tp, _Up > >, is_constructible< _Tp, const _Up & >, is_convertible< const _Up &, _Tp >, __not_< __converts_from_optional< _Tp, _Up > > >::value, bool > = true>
  constexpr **optional** (const optional< _Up > &__t)
- template<typename _Up, enable_if_t< __and_< __not_< is_same< _Tp, _Up > >, is_constructible< _Tp, const _Up & >, __not_< is_convertible< const _Up &, _Tp > >, __not_< __converts_from_optional< _Tp, _Up > > >::value, bool > = false>
  constexpr **optional** (const optional< _Up > &__t)
- template<typename _Up, enable_if_t< __and_< __not_< is_same< _Tp, _Up > >, is_constructible< _Tp, _Up && >, is_convertible< _Up &&, _Tp >, __not_< __converts_from_optional< _Tp, _Up > > >::value, bool > = true>
  constexpr **optional** (optional< _Up > &&__t)

- template<typename _Up, enable_if_t< __and_< __not_< is_same< _Tp, _Up > >, is_constructible< _Tp, _Up && >, __not_< is_convertible< _Up &&, _Tp > >, __not_< __converts_from_optional< _Tp, _Up > > >::value, bool > = false>
  constexpr **optional** (optional< _Up > &&__t)
- template<typename... _Args>
  enable_if_t< is_constructible< _Tp, _Args &&... >::value > **emplace** (_Args &&... __args)
- template<typename _Up, typename... _Args>
  enable_if_t< is_constructible< _Tp, initializer_list< _Up > &, _Args &&... >::value > **emplace** (initializer_list< _Up > __il, _Args &&... __args)
- constexpr **operator bool** () const noexcept
- constexpr _Tp & **operator*** () &
- constexpr _Tp && **operator*** () &&
- constexpr const _Tp & **operator*** () const &
- constexpr const _Tp && **operator*** () const &&
- _Tp * **operator->** ()
- constexpr const _Tp * **operator->** () const
- template<typename _Up = _Tp>
  enable_if_t< __and_< __not_< is_same< optional< _Tp >, decay_t< _Up > > >, is_constructible< _Tp, _Up >, __not_< __and_< is_scalar< _Tp >, is_same< _Tp, decay_t< _Up > > >, is_assignable< _Tp &, _Up > >::value, optional & > **operator=** (_Up &&__u)
- template<typename _Up>
  enable_if_t< __and_< __not_< is_same< _Tp, _Up > >, is_constructible< _Tp, const _Up & >, is_assignable< _Tp &, _Up >, __not_< __converts_from_optional< _Tp, _Up > >, __not_< __assigns_from_optional< _Tp, _Up > > >::value, optional & > **operator=** (const optional< _Up > &__u)
- optional & **operator=** (nullopt_t) noexcept
- template<typename _Up>
  enable_if_t< __and_< __not_< is_same< _Tp, _Up > >, is_constructible< _Tp, _Up >, is_assignable< _Tp &, _Up >, __not_< __converts_from_optional< _Tp, _Up > >, __not_< __assigns_from_optional< _Tp, _Up > > >::value, optional & > **operator=** (optional< _Up > &&__u)
- void **swap** (optional &__other) noexcept(is_nothrow_move_constructible< _Tp >() &&__is_nothrow_swappable< _Tp >::value)
- constexpr _Tp & **value** () &
- constexpr _Tp && **value** () &&
- constexpr const _Tp & **value** () const &
- constexpr const _Tp && **value** () const &&
- template<typename _Up>
  _Tp **value_or** (_Up &&__u) &&
- template<typename _Up>
  constexpr _Tp **value_or** (_Up &&__u) const &

5.734.1 Detailed Description

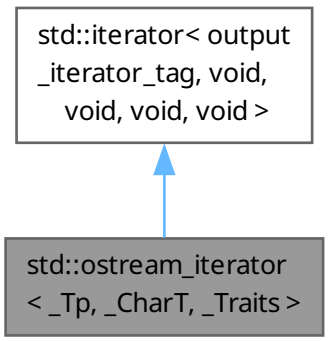
template<typename _Tp>
class std::experimental::fundamentals_v1::optional< _Tp >

Class template for optional values.
The documentation for this class was generated from the following file:

- experimental/optional

5.735 std::ostream_iterator< _Tp, _CharT, _Traits > Class Template Reference

`#include <stream_iterator.h>`
Inheritance diagram for std::ostream_iterator< _Tp, _CharT, _Traits >:



Public Types

- typedef output_iterator_tag iterator_category
- typedef void pointer
- typedef void reference
- typedef void value_type

- using difference_type
- typedef _CharT char_type
- typedef _Traits traits_type
- typedef basic_ostream< _CharT, _Traits > ostream_type

Public Member Functions

- ostream_iterator (const ostream_iterator &__obj) noexcept
- ostream_iterator (ostream_type &__s) noexcept
- ostream_iterator (ostream_type &__s, const _CharT *__c) noexcept
- ostream_iterator & **operator*** () noexcept
- ostream_iterator & **operator++** () noexcept
- ostream_iterator & **operator++** (int) noexcept
- ostream_iterator & operator= (const _Tp &__value)
- ostream_iterator & **operator=** (const ostream_iterator &)=default

5.735.1 Detailed Description

template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>
class std::ostream_iterator< _Tp, _CharT, _Traits >

Provides output iterator semantics for streams.
This class provides an iterator to write to an ostream. The type Tp is the only type written by this iterator and there must be an operator<<(Tp) defined.

Template Parameters

_Tp	The type to write to the ostream.
_CharT	The ostream char_type.
_Traits	The ostream char_traits.

5.735.2 Member Typedef Documentation

char_type

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>
typedef _CharT std::ostream_iterator< _Tp, _CharT, _Traits >::char_type
```

Public typedef.

difference_type

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>
using std::ostream_iterator< _Tp, _CharT, _Traits >::difference_type
```

Public typedef.

iterator_category

```
typedef output_iterator_tag std::iterator< output_iterator_tag, void, void, void, void >::iterator↩
_category  [inherited]
```

One of the tag types.

ostream_type

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>
typedef basic_ostream<_CharT, _Traits> std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_↩
type
```

Public typedef.

pointer

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::pointer  [inherited]
```

This type represents a pointer-to-value_type.

reference

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::reference  [inherited]
```

This type represents a reference-to-value_type.

traits_type

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>
typedef _Traits std::ostream_iterator< _Tp, _CharT, _Traits >::traits_type
```

Public typedef.

value_type

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::value_type  [inherited]
```

The type "pointed to" by the iterator.

5.735.3 Constructor & Destructor Documentation

ostream_iterator() [1/3]

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>
std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator (
            ostream_type & __s)  [inline], [noexcept]
```

Construct from an ostream.
References std::__addressof().
Referenced by ostream_iterator().

ostream_iterator() [2/3]

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>
std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator (
            ostream_type & __s,
            const _CharT * __c)  [inline], [noexcept]
```

Construct from an ostream.
The delimiter string *c* is written to the stream after every Tp written to the stream. The delimiter is not copied, and thus must not be destroyed while this iterator is in use.

Parameters

_↩ _s	Underlying ostream to write to.
_↩ _c	CharT delimiter string to insert.

References std::__addressof().

ostream_iterator() [3/3]

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>
std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator (
            const ostream_iterator< _Tp, _CharT, _Traits > & __obj)  [inline], [noexcept]
```

Copy constructor.
References ostream_iterator().

5.735.4 Member Function Documentation

operator=()

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>
ostream_iterator & std::ostream_iterator< _Tp, _CharT, _Traits >::operator= (
            const _Tp & __value)  [inline]
```

Writes *value* to underlying ostream using operator<<. If constructed with delimiter string, writes delimiter to ostream.
The documentation for this class was generated from the following file:

- stream_iterator.h

5.736 std::experimental::fundamentals_v2::ostream_joiner< _DelimT, _CharT, _Traits > Class Template Reference

```
#include <iterator>
```

Public Types

- typedef _CharT **char_type**
- typedef void **difference_type**
- typedef output_iterator_tag **iterator_category**
- typedef basic_ostream< _CharT, _Traits > **ostream_type**
- typedef void **pointer**
- typedef void **reference**
- typedef _Traits **traits_type**
- typedef void **value_type**

Public Member Functions

- **ostream_joiner** (ostream_type &__os, _DelimT &&__delimiter) noexcept(is_nothrow_move_constructible_v< _DelimT >)
- **ostream_joiner** (ostream_type &__os, const _DelimT &__delimiter) noexcept(is_nothrow_copy_constructible_v< _DelimT >)
- ostream_joiner & **operator*** () noexcept
- ostream_joiner & **operator++** () noexcept
- ostream_joiner & **operator++** (int) noexcept
- template<typename _Tp>
  ostream_joiner & **operator=** (const _Tp &__value)

5.736.1 Detailed Description

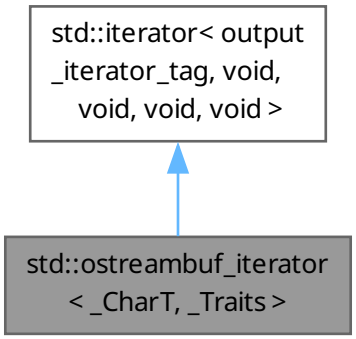
template<typename _DelimT, typename _CharT = char, typename _Traits = char_traits<_CharT>>
class std::experimental::fundamentals_v2::ostream_joiner< _DelimT, _CharT, _Traits >

Output iterator that inserts a delimiter between elements.
The documentation for this class was generated from the following file:

- experimental/iterator

5.737 std::ostreambuf_iterator< _CharT, _Traits > Class Template Reference

`#include <streambuf_iterator.h>`
Inheritance diagram for std::ostreambuf_iterator< _CharT, _Traits >:



Public Types

- typedef output_iterator_tag iterator_category
- typedef void pointer
- typedef void reference
- typedef void value_type

- using difference_type
- typedef _CharT char_type
- typedef _Traits traits_type
- typedef basic_streambuf< _CharT, _Traits > streambuf_type
- typedef basic_ostream< _CharT, _Traits > ostream_type

Public Member Functions

- ostreambuf_iterator (ostream_type &__s) noexcept
- ostreambuf_iterator (streambuf_type *__s) noexcept
- ostreambuf_iterator & **_M_put** (const _CharT *__ws, streamsize __len)
- bool failed () const noexcept
- ostreambuf_iterator & operator* ()
- ostreambuf_iterator & operator++ ()
- ostreambuf_iterator & operator++ (int)
- ostreambuf_iterator & operator= (_CharT __c)

Friends

- template<typename _CharT2>
  __gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, ostreambuf_iterator< _CharT2 > >::__type **copy** (istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, ostreambuf_iterator< _CharT2 >)

5.737.1 Detailed Description

template<typename _CharT, typename _Traits>
class std::ostreambuf_iterator< _CharT, _Traits >

Provides output iterator semantics for streambufs.

5.737.2 Member Typedef Documentation

char_type

```
template<typename _CharT, typename _Traits>
typedef _CharT std::ostreambuf_iterator< _CharT, _Traits >::char_type
```

Public typedefs.

difference_type

```
template<typename _CharT, typename _Traits>
using std::ostreambuf_iterator< _CharT, _Traits >::difference_type
```

Public typedefs.

iterator_category

```
typedef output_iterator_tag std::iterator< output_iterator_tag, void, void, void, void >::iterator↩
_category [inherited]
```

One of the tag types.

ostream_type

```
template<typename _CharT, typename _Traits>
typedef basic_ostream<_CharT, _Traits> std::ostreambuf_iterator< _CharT, _Traits >::ostream_type
```
Public typedefs.

pointer

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::pointer  [inherited]
```
This type represents a pointer-to-value_type.

reference

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::reference  [inherited]
```
This type represents a reference-to-value_type.

streambuf_type

```
template<typename _CharT, typename _Traits>
typedef basic_streambuf<_CharT, _Traits> std::ostreambuf_iterator< _CharT, _Traits >::streambuf↩
_type
```
Public typedefs.

traits_type

```
template<typename _CharT, typename _Traits>
typedef _Traits std::ostreambuf_iterator< _CharT, _Traits >::traits_type
```
Public typedefs.

value_type

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::value_type  [inherited]
```
The type "pointed to" by the iterator.

5.737.3 Constructor & Destructor Documentation

ostreambuf_iterator() [1/2]

```
template<typename _CharT, typename _Traits>
std::ostreambuf_iterator< _CharT, _Traits >::ostreambuf_iterator (
            ostream_type & __s)  [inline], [noexcept]
```
Construct output iterator from ostream.

ostreambuf_iterator() [2/2]

```
template<typename _CharT, typename _Traits>
std::ostreambuf_iterator< _CharT, _Traits >::ostreambuf_iterator (
            streambuf_type * __s)  [inline], [noexcept]
```
Construct output iterator from streambuf.

5.737.4 Member Function Documentation

failed()

```
template<typename _CharT, typename _Traits>
bool std::ostreambuf_iterator< _CharT, _Traits >::failed () const  [inline], [nodiscard], [noexcept]
```
Return true if previous operator=() failed.

Referenced by std::basic_ostream< _CharT, _Traits >::operator<<().

operator∗()

```
template<typename _CharT, typename _Traits>
ostreambuf_iterator & std::ostreambuf_iterator< _CharT, _Traits >::operator* ()  [inline], [nodiscard]
```

Return ∗this.

operator++() [1/2]

```
template<typename _CharT, typename _Traits>
ostreambuf_iterator & std::ostreambuf_iterator< _CharT, _Traits >::operator++ ()  [inline]
```

Return ∗this.

operator++() [2/2]

```
template<typename _CharT, typename _Traits>
ostreambuf_iterator & std::ostreambuf_iterator< _CharT, _Traits >::operator++ (
            int )  [inline]
```

Return ∗this.

operator=()

```
template<typename _CharT, typename _Traits>
ostreambuf_iterator & std::ostreambuf_iterator< _CharT, _Traits >::operator= (
            _CharT __c)  [inline]
```

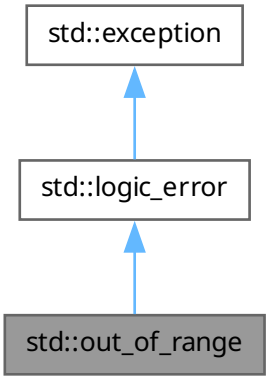
Write character to streambuf. Calls streambuf.sputc().
The documentation for this class was generated from the following files:

- iosfwd
- streambuf_iterator.h

5.738 std::out_of_range Class Reference

```
#include <stdexcept>
```

Inheritance diagram for std::out_of_range:



Public Member Functions

- **out_of_range** (const char *)
- **out_of_range** (const out_of_range &)=default
- **out_of_range** (const string &__arg)
- **out_of_range** (out_of_range &&)=default
- out_of_range & **operator=** (const out_of_range &)=default
- out_of_range & **operator=** (out_of_range &&)=default
- virtual const char * what () const noexcept

5.738.1 Detailed Description

This represents an argument whose value is not within the expected range (e.g., boundary checks in basic_string).

5.738.2 Member Function Documentation

what()

```
virtual const char * std::logic_error::what () const  [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).
Reimplemented from std::exception.
Reimplemented in std::future_error.
The documentation for this class was generated from the following file:

- stdexcept

5.739 std::output_iterator_tag Struct Reference

```
#include <stl_iterator_base_types.h>
```

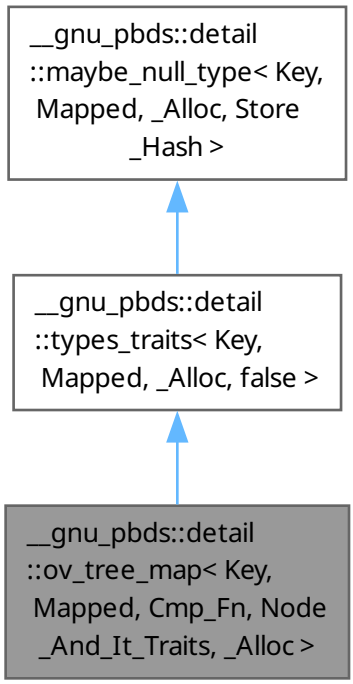
5.739.1 Detailed Description

Marking output iterators.
The documentation for this struct was generated from the following file:

- stl_iterator_base_types.h

5.740 __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > Class Template Reference

`#include <ov_tree_map_.hpp>`
Inheritance diagram for __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >:



Classes

- class cond_dtor

Public Types

- typedef _Alloc **allocator_type**
- typedef Cmp_Fn **cmp_fn**
- typedef std::pair< size_type, size_type > **comp_hash**
- typedef point_const_iterator **const_iterator**
- typedef traits_base::const_pointer **const_pointer**
- typedef traits_base::const_reference **const_reference**

- typedef ov_tree_tag **container_category**
- typedef _Alloc::difference_type **difference_type**
- typedef point_iterator **iterator**
- typedef traits_base::key_const_pointer **key_const_pointer**
- typedef traits_base::key_const_reference **key_const_reference**
- typedef traits_base::key_pointer **key_pointer**
- typedef traits_base::key_reference **key_reference**
- typedef traits_base::key_type **key_type**
- typedef traits_base::mapped_const_pointer **mapped_const_pointer**
- typedef traits_base::mapped_const_reference **mapped_const_reference**
- typedef traits_base::mapped_pointer **mapped_pointer**
- typedef traits_base::mapped_reference **mapped_reference**
- typedef traits_base::mapped_type **mapped_type**
- typedef __nothrowcopy::indicator **no_throw_indicator**
- typedef traits_type::node_const_iterator **node_const_iterator**
- typedef traits_type::node_iterator **node_iterator**
- typedef traits_type::node_update **node_update**
- typedef const_pointer **point_const_iterator**
- typedef pointer **point_iterator**
- typedef traits_base::pointer **pointer**
- typedef traits_base::reference **reference**
- typedef _Alloc::size_type **size_type**
- typedef integral_constant< int, Store_Hash > **store_extra**
- typedef stored_data< value_type, size_type, Store_Hash > **stored_data_type**
- typedef traits_base::value_type **value_type**

Public Member Functions

- **ov_tree_map** (const Cmp_Fn &)
- **ov_tree_map** (const Cmp_Fn &, const node_update &)
- **ov_tree_map** (const ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- template<typename It>
  void **copy_from_range** (It, It)
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- iterator **erase** (iterator it)
- bool **erase** (key_const_reference)
- template<typename Pred>
  size_type **erase_if** (Pred)
- point_iterator **find** (key_const_reference r_key)
- point_const_iterator **find** (key_const_reference r_key) const
- Cmp_Fn & **get_cmp_fn** ()
- const Cmp_Fn & **get_cmp_fn** () const
- std::pair< point_iterator, bool > **insert** (const_reference r_value)
- void **join** (ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- point_iterator **lower_bound** (key_const_reference r_key)
- point_const_iterator **lower_bound** (key_const_reference r_key) const

- size_type **max_size** () const
- node_iterator node_begin ()
- node_const_iterator node_begin () const
- node_iterator node_end ()
- node_const_iterator node_end () const
- mapped_reference **operator[ ]** (key_const_reference r_key)
- size_type **size** () const
- void **split** (key_const_reference, ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- void **swap** (ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- point_iterator **upper_bound** (key_const_reference r_key)
- point_const_iterator **upper_bound** (key_const_reference r_key) const

Public Attributes

- no_throw_indicator **m_no_throw_copies_indicator**
- no_throw_indicator **m_no_throw_copies_indicator**
- store_extra **m_store_extra_indicator**
- store_extra **m_store_extra_indicator**

5.740.1 Detailed Description

template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _↩ Alloc>
class __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >

Ordered-vector tree associative-container.

5.740.2 Member Function Documentation

node_begin() [1/2]

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _↩
Alloc>
ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator __gnu_pbds::detail::ov_tree_map<
Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin ()  [inline]
```

Returns a node_iterator corresponding to the node at the root of the tree.

node_begin() [2/2]

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _↩
Alloc>
ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __gnu_pbds::detail::ov_tree_m
Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin () const  [inline]
```

Returns a const node_iterator corresponding to the node at the root of the tree.

node_end() [1/2]

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _↩
Alloc>
ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator __gnu_pbds::detail::ov_tree_map<
Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end ()  [inline]
```

Returns a node_iterator corresponding to a node just after a leaf of the tree.

node_end() [2/2]

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _↩
Alloc>
ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __gnu_pbds::detail::ov_tree_n
Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end () const  [inline]
```

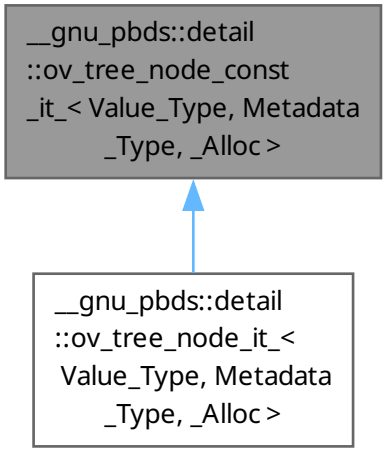
Returns a const node_iterator corresponding to a node just after a leaf of the tree.
The documentation for this class was generated from the following file:

- ov_tree_map_.hpp

5.741 __gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc > Class Template Reference

```
#include <node_iterators.hpp>
```

Inheritance diagram for __gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >:



Public Types

- typedef rebind_traits< _Alloc, typenameremove_const< Value_Type >::type >::const_pointer **const_reference**
- typedef trivial_iterator_difference_type **difference_type**
- typedef trivial_iterator_tag **iterator_category**
- typedef rebind_traits< _Alloc, metadata_type >::const_reference **metadata_const_reference**
- typedef Metadata_Type **metadata_type**
- typedef rebind_traits< _Alloc, typenameremove_const< Value_Type >::type >::const_pointer **reference**
- typedef rebind_traits< _Alloc, Value_Type >::const_pointer **value_type**

Public Member Functions

- **ov_tree_node_const_it_** (const_pointer p_nd=0, const_pointer p_begin_nd=0, const_pointer p_end_nd=0, const_metadata_pointer p_metadata=0)
- this_type get_l_child () const

- metadata_const_reference **get_metadata** () const
- this_type get_r_child () const
- bool **operator!=** (const this_type &other) const
- const_reference **operator*** () const
- bool **operator==** (const this_type &other) const

Public Attributes

- pointer **m_p_begin_value**
- pointer **m_p_end_value**
- const_metadata_pointer **m_p_metadata**
- pointer **m_p_value**

Protected Types

- typedef rebind_traits< _Alloc, Metadata_Type >::const_pointer **const_metadata_pointer**
- typedef rebind_traits< _Alloc, Value_Type >::const_pointer **const_pointer**
- typedef rebind_traits< _Alloc, Value_Type >::pointer **pointer**
- typedef ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc > **this_type**

Static Protected Member Functions

- template<typename Ptr>
  static Ptr **mid_pointer** (Ptr p_begin, Ptr p_end)

5.741.1 Detailed Description

template<typename Value_Type, typename Metadata_Type, typename _Alloc>
class __gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >

Const node reference.

5.741.2 Member Function Documentation

get_l_child()

```
template<typename Value_Type, typename Metadata_Type, typename _Alloc>
this_type __gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >::get_↩
l_child () const  [inline]
```

Returns the node iterator associated with the left node.

get_r_child()

```
template<typename Value_Type, typename Metadata_Type, typename _Alloc>
this_type __gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >::get_↩
r_child () const  [inline]
```

Returns the node iterator associated with the right node.

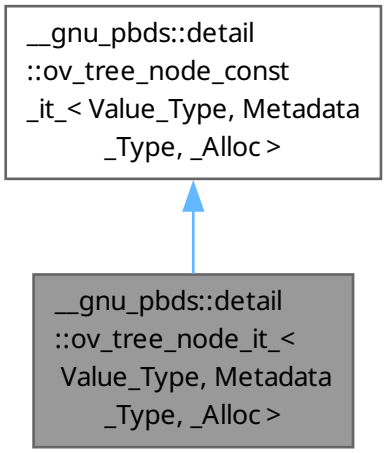
The documentation for this class was generated from the following file:

- ov_tree_map_/node_iterators.hpp

5.742 __gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc > Class Template Reference

`#include <node_iterators.hpp>`

Inheritance diagram for __gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >:



Public Types

- typedef rebind_traits< _Alloc, typenameremove_const< Value_Type >::type >::pointer **const_reference**
- typedef trivial_iterator_difference_type **difference_type**
- typedef trivial_iterator_tag **iterator_category**
- typedef rebind_traits< _Alloc, metadata_type >::const_reference **metadata_const_reference**
- typedef Metadata_Type **metadata_type**
- typedef rebind_traits< _Alloc, typenameremove_const< Value_Type >::type >::pointer **reference**
- typedef rebind_traits< _Alloc, Value_Type >::pointer **value_type**

Public Member Functions

- **ov_tree_node_it_** (const_pointer p_nd=0, const_pointer p_begin_nd=0, const_pointer p_end_nd=0, const_↩ metadata_pointer p_metadata=0)
- ov_tree_node_it_ get_l_child () const
- metadata_const_reference **get_metadata** () const
- ov_tree_node_it_ get_r_child () const
- bool **operator!=** (const this_type &other) const
- reference operator∗ () const
- bool **operator==** (const this_type &other) const

Public Attributes

- pointer **m_p_begin_value**
- pointer **m_p_end_value**
- const_metadata_pointer **m_p_metadata**
- pointer **m_p_value**

Static Protected Member Functions

- template<typename Ptr>
  static Ptr **mid_pointer** (Ptr p_begin, Ptr p_end)

5.742.1 Detailed Description

template<typename Value_Type, typename Metadata_Type, typename _Alloc>
class __gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >

Node reference.

5.742.2 Member Function Documentation

get_l_child()

```
template<typename Value_Type, typename Metadata_Type, typename _Alloc>
ov_tree_node_it_ __gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >::get↩
_l_child () const  [inline]
```

Returns the node reference associated with the left node.

get_r_child()

```
template<typename Value_Type, typename Metadata_Type, typename _Alloc>
ov_tree_node_it_ __gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >::get↩
_r_child () const  [inline]
```

Returns the node reference associated with the right node.

operator*()

```
template<typename Value_Type, typename Metadata_Type, typename _Alloc>
reference __gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >::operator* ()
const  [inline]
```

Access.

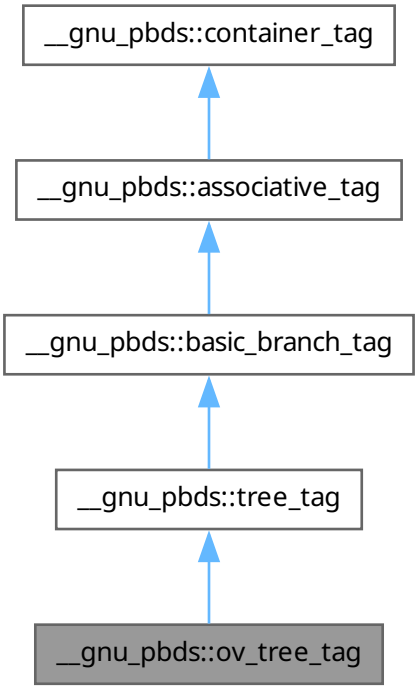
The documentation for this class was generated from the following file:

- ov_tree_map_/node_iterators.hpp

5.743 __gnu_pbds::ov_tree_tag Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for __gnu_pbds::ov_tree_tag:



5.743.1 Detailed Description

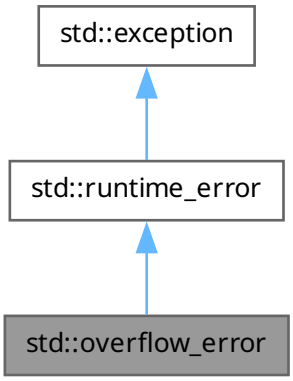
Ordered-vector tree.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

5.744 std::overflow_error Class Reference

`#include <stdexcept>`

Inheritance diagram for std::overflow_error:



Public Member Functions

- **overflow_error** (const char *)
- **overflow_error** (const overflow_error &)=default
- **overflow_error** (const string &__arg)
- **overflow_error** (overflow_error &&)=default
- overflow_error & **operator=** (const overflow_error &)=default
- overflow_error & **operator=** (overflow_error &&)=default
- virtual const char * what () const noexcept

5.744.1 Detailed Description

Thrown to indicate arithmetic overflow.

5.744.2 Member Function Documentation

what()

```
virtual const char * std::runtime_error::what () const  [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).
Reimplemented from std::exception.
Reimplemented in std::experimental::filesystem::v1::filesystem_error, and std::filesystem::filesystem_error.
The documentation for this class was generated from the following file:

- stdexcept

5.745 std::owner_less< _Tp > Struct Template Reference

5.745.1 Detailed Description

template<typename _Tp = void>
struct std::owner_less< _Tp >

Primary template owner_less.

The documentation for this struct was generated from the following file:

- bits/shared_ptr.h

5.746 std::experimental::fundamentals_v2::owner_less< shared_ptr< _Tp > > Struct Template Reference

`#include <shared_ptr.h>`

Public Types

- typedef _Tp first_argument_type
- typedef bool result_type
- typedef _Tp second_argument_type

Public Member Functions

- bool **operator()** (const _Tp &__lhs, const _Tp &__rhs) const noexcept
- bool **operator()** (const _Tp &__lhs, const _Tp1 &__rhs) const noexcept
- bool **operator()** (const _Tp1 &__lhs, const _Tp &__rhs) const noexcept

5.746.1 Detailed Description

template<typename _Tp>
struct std::experimental::fundamentals_v2::owner_less< shared_ptr< _Tp > >

Partial specialization of owner_less for shared_ptr.

5.746.2 Member Typedef Documentation

first_argument_type

`typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type [inherited]`
`first_argument_type` is the type of the first argument

result_type

`typedef bool std::binary_function< _Tp, _Tp, bool >::result_type [inherited]`
`result_type` is the return type

second_argument_type

`typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type [inherited]`
`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- experimental/bits/shared_ptr.h

5.747 std::owner_less< shared_ptr< _Tp > > Struct Template Reference

`#include <shared_ptr.h>`

Public Types

- typedef _Tp first_argument_type
- typedef bool result_type
- typedef _Tp second_argument_type

Public Member Functions

- bool **operator()** (const _Tp &__lhs, const _Tp &__rhs) const noexcept
- bool **operator()** (const _Tp &__lhs, const _Tp1 &__rhs) const noexcept
- bool **operator()** (const _Tp1 &__lhs, const _Tp &__rhs) const noexcept

5.747.1 Detailed Description

template<typename _Tp>
struct std::owner_less< shared_ptr< _Tp > >

Partial specialization of owner_less for shared_ptr.

5.747.2 Member Typedef Documentation

first_argument_type

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type  [inherited]
```

`first_argument_type` is the type of the first argument

result_type

```
typedef bool std::binary_function< _Tp, _Tp, bool >::result_type  [inherited]
```

`result_type` is the return type

second_argument_type

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type  [inherited]
```

`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- bits/shared_ptr.h

5.748 std::owner_less< void > Struct Reference

`#include <shared_ptr.h>`

Public Types

- typedef _Tp first_argument_type
- typedef bool result_type
- typedef _Tp second_argument_type

Public Member Functions

- bool **operator()** (const _Tp &__lhs, const _Tp &__rhs) const noexcept
- bool **operator()** (const _Tp &__lhs, const _Tp1 &__rhs) const noexcept
- bool **operator()** (const _Tp1 &__lhs, const _Tp &__rhs) const noexcept

5.748.1 Detailed Description

Void specialization of owner_less compares either shared_ptr or weak_ptr.

5.748.2 Member Typedef Documentation

first_argument_type

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type  [inherited]
```
first_argument_type is the type of the first argument

result_type

```
typedef bool std::binary_function< _Tp, _Tp, bool >::result_type  [inherited]
```
result_type is the return type

second_argument_type

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type  [inherited]
```
second_argument_type is the type of the second argument
The documentation for this struct was generated from the following file:

- bits/shared_ptr.h

5.749 std::experimental::fundamentals_v2::owner_less< weak_ptr< _Tp > > Struct Template Reference

```
#include <shared_ptr.h>
```

Public Types

- typedef _Tp first_argument_type
- typedef bool result_type
- typedef _Tp second_argument_type

Public Member Functions

- bool **operator()** (const _Tp &__lhs, const _Tp &__rhs) const noexcept
- bool **operator()** (const _Tp &__lhs, const _Tp1 &__rhs) const noexcept
- bool **operator()** (const _Tp1 &__lhs, const _Tp &__rhs) const noexcept

5.749.1 Detailed Description

template<typename _Tp>
struct std::experimental::fundamentals_v2::owner_less< weak_ptr< _Tp > >

Partial specialization of owner_less for weak_ptr.

5.749.2 Member Typedef Documentation

first_argument_type

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type  [inherited]
```
first_argument_type is the type of the first argument

result_type

```
typedef bool std::binary_function< _Tp, _Tp, bool >::result_type  [inherited]
```
result_type is the return type

second_argument_type

typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type [inherited]
second_argument_type is the type of the second argument
The documentation for this struct was generated from the following file:

- experimental/bits/shared_ptr.h

5.750 std::owner_less< weak_ptr< _Tp > > Struct Template Reference

#include <shared_ptr.h>

Public Types

- typedef _Tp first_argument_type
- typedef bool result_type
- typedef _Tp second_argument_type

Public Member Functions

- bool **operator()** (const _Tp &__lhs, const _Tp &__rhs) const noexcept
- bool **operator()** (const _Tp &__lhs, const _Tp1 &__rhs) const noexcept
- bool **operator()** (const _Tp1 &__lhs, const _Tp &__rhs) const noexcept

5.750.1 Detailed Description

template<typename _Tp>
struct std::owner_less< weak_ptr< _Tp > >

Partial specialization of owner_less for weak_ptr.

5.750.2 Member Typedef Documentation

first_argument_type

typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type [inherited]
first_argument_type is the type of the first argument

result_type

typedef bool std::binary_function< _Tp, _Tp, bool >::result_type [inherited]
result_type is the return type

second_argument_type

typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type [inherited]
second_argument_type is the type of the second argument
The documentation for this struct was generated from the following file:

- bits/shared_ptr.h

5.751 std::packaged_task< _Res(_ArgTypes...)> Class Template Reference

#include <future>

Public Member Functions

- template<typename _Fn, typename = __not_same<_Fn>>
  packaged_task (_Fn &&__fn)
- **packaged_task** (const packaged_task &)=delete
- **packaged_task** (packaged_task &&__other) noexcept
- future< _Res > **get_future** ()
- void **make_ready_at_thread_exit** (_ArgTypes... __args)
- void **operator()** (_ArgTypes... __args)
- packaged_task & **operator=** (const packaged_task &)=delete
- packaged_task & **operator=** (packaged_task &&__other) noexcept
- void **reset** ()
- void **swap** (packaged_task &__other) noexcept
- bool **valid** () const noexcept

5.751.1 Detailed Description

template<typename _Res, typename... _ArgTypes>
class std::packaged_task< _Res(_ArgTypes...)>

packaged_task
The documentation for this class was generated from the following file:

- future

5.752 __gnu_cxx::pair< _T1, _T2 > Struct Template Reference

`#include <utility>`

Public Types

- typedef _T1 first_type
- typedef _T2 second_type

Public Member Functions

- template<typename _U1 = _T1, typename _U2 = _T2, typename enable_if< __and_< __is_implicitly_default_constructible< _U1 >, __↩
  is_implicitly_default_constructible< _U2 > > ::value, bool >::type = true>
  constexpr pair ()
- template<typename _U2, __enable_if_t< __and_< is_pointer< _T1 >, __not_< is_reference< _U2 > >, is_constructible< _T2, _U2 >, __not_< is_constructible< _T2, const _U2 & > >, is_convertible< _U2, _T2 > >::value, bool > = true>
  constexpr **pair** (__zero_as_null_pointer_constant, _U2 &&__y,...)
- template<typename _U2, __enable_if_t< __and_< is_pointer< _T1 >, __not_< is_reference< _U2 > >, is_constructible< _T2, _U2 >, __not_< is_constructible< _T2, const _U2 & > >, __not_< is_convertible< _U2, _T2 > > >::value, bool > = false>
  constexpr **pair** (__zero_as_null_pointer_constant, _U2 &&__y,...)
- template<typename _U1, __enable_if_t< __and_< __not_< is_reference< _U1 > >, is_pointer< _T2 >, is_constructible< _T1, _U1 >, __not_< is_constructible< _T1, const _U1 & > >, is_convertible< _U1, _T1 > >::value, bool > = true>
  constexpr **pair** (_U1 &&__x, __zero_as_null_pointer_constant,...)
- template<typename _U1, __enable_if_t< __and_< __not_< is_reference< _U1 > >, is_pointer< _T2 >, is_constructible< _T1, _U1 >, __not_< is_constructible< _T1, const _U1 & > >, __not_< is_convertible< _U1, _T1 > > >::value, bool > = false>
  constexpr **pair** (_U1 &&__x, __zero_as_null_pointer_constant,...)
- template<typename _U1, typename _U2, typename enable_if< _PCCP::template _MoveConstructiblePair< _U1, _U2 >() && _PCCP↩
  ::template _ImplicitlyMoveConvertiblePair< _U1, _U2 >(), bool >::type = true>
  constexpr **pair** (_U1 &&__x, _U2 &&__y)

- template<typename _U1, typename _U2, typename enable_if< _PCCP::template _MoveConstructiblePair< _U1, _U2 >() &&!_PCCP::template _ImplicitlyMoveConvertiblePair< _U1, _U2 >(), bool >::type = false>
  constexpr **pair** (_U1 &&__x, _U2 &&__y)
- template<typename _U1 = _T1, typename _U2 = _T2, typename enable_if< _PCCP::template _ConstructiblePair< _U1, _U2 >() &&_PCCP::template _ImplicitlyConvertiblePair< _U1, _U2 >(), bool >::type = true>
  constexpr pair (const _T1 &__a, const _T2 &__b)
- template<typename _U1 = _T1, typename _U2 = _T2, typename enable_if< _PCCP::template _ConstructiblePair< _U1, _U2 >() &&!_PCCP::template _ImplicitlyConvertiblePair< _U1, _U2 >(), bool >::type = false>
  constexpr pair (const _T1 &__a, const _T2 &__b)
- constexpr pair (const pair &)=default
- template<typename _U1, typename _U2, typename enable_if< _PCCFP< _U1, _U2 >::template _ConstructiblePair< _U1, _U2 >() &&_PCCFP< _U1, _U2 >::template _ImplicitlyConvertiblePair< _U1, _U2 >(), bool >::type = true>
  constexpr **pair** (const pair< _U1, _U2 > &__p)
- template<typename _U1, typename _U2, typename enable_if< _PCCFP< _U1, _U2 >::template _ConstructiblePair< _U1, _U2 >() &&!_PCCFP< _U1, _U2 >::template _ImplicitlyConvertiblePair< _U1, _U2 >(), bool >::type = false>
  constexpr **pair** (const pair< _U1, _U2 > &__p)
- constexpr pair (pair &&)=default
- template<typename _U1, typename _U2, typename enable_if< _PCCFP< _U1, _U2 >::template _MoveConstructiblePair< _U1, _U2 >() &&_PCCFP< _U1, _U2 >::template _ImplicitlyMoveConvertiblePair< _U1, _U2 >(), bool >::type = true>
  constexpr **pair** (pair< _U1, _U2 > &&__p)
- template<typename _U1, typename _U2, typename enable_if< _PCCFP< _U1, _U2 >::template _MoveConstructiblePair< _U1, _U2 >() &&!_PCCFP< _U1, _U2 >::template _ImplicitlyMoveConvertiblePair< _U1, _U2 >(), bool >::type = false>
  constexpr **pair** (pair< _U1, _U2 > &&__p)
- template<typename... _Args1, typename... _Args2>
  constexpr pair (piecewise_construct_t, tuple< _Args1... >, tuple< _Args2... >)
- pair & **operator=** (__conditional_t< __and_< is_copy_assignable< _T1 >, is_copy_assignable< _T2 > >::value, const pair &, const __nonesuch & > __p)
- pair & **operator=** (__conditional_t< __and_< is_move_assignable< _T1 >, is_move_assignable< _T2 > >::value, pair &&, __nonesuch && > __p) noexcept(__and_< is_nothrow_move_assignable< _T1 >, is_nothrow_move_assignable< _T2 > >::value)
- template<typename _U1, typename _U2>
  enable_if< __and_< is_assignable< _T1 &, const _U1 & >, is_assignable< _T2 &, const _U2 & > >::value, pair & >::type **operator=** (const pair< _U1, _U2 > &__p)
- template<typename _U1, typename _U2>
  enable_if< __and_< is_assignable< _T1 &, _U1 && >, is_assignable< _T2 &, _U2 && > >::value, pair & >::type **operator=** (pair< _U1, _U2 > &&__p)
- constexpr void swap (pair &__p) noexcept(__and_< __is_nothrow_swappable< _T1 >, __is_nothrow_swappable< _T2 > >::value)

Public Attributes

- _T1 first
- _T2 second

Related Symbols

(Note that these are not member symbols.)

- template<typename _T1, typename _T2>
  pair (_T1, _T2) -> pair< _T1, _T2 >
- template<typename _T1, typename _T2>
  constexpr enable_if< __and_< __is_swappable< _T1 >, __is_swappable< _T2 > >::value >::type swap (pair< _T1, _T2 > &__x, pair< _T1, _T2 > &__y) noexcept(noexcept(__x.swap(__y)))

- template<typename _T1, typename _T2, typename _U1, typename _U2>
  constexpr bool operator== (const pair< _T1, _T2 > &__x, const pair< _U1, _U2 > &__y)
- template<typename _T1, typename _T2, typename _U1, typename _U2>
  constexpr common_comparison_category_t< __detail::__synth3way_t< _T1, _U1 >, __detail::__synth3way_t< _T2, _U2 > > operator<=> (const pair< _T1, _T2 > &__x, const pair< _U1, _U2 > &__y)

5.752.1 Detailed Description

template<typename _T1, typename _T2>
struct __gnu_cxx::pair< _T1, _T2 >

Struct holding two objects of arbitrary type.

Template Parameters

_T1	Type of first object.
_T2	Type of second object.

https://gcc.gnu.org/onlinedocs/libstdc++/manual/utilities.html

5.752.2 Member Typedef Documentation

first_type

```
template<typename _T1, typename _T2>
typedef _T1 std::pair< _T1, _T2 >::first_type
```

The type of the `first` member.

second_type

```
template<typename _T1, typename _T2>
typedef _T2 std::pair< _T1, _T2 >::second_type
```

The type of the `second` member.

5.752.3 Constructor & Destructor Documentation

pair() [1/5]

```
template<typename _T1, typename _T2>
std::pair< _T1, _T2 >::pair (
            const pair< _T1, _T2 > & )  [constexpr], [default]
```

Copy constructor.

pair() [2/5]

```
template<typename _T1, typename _T2>
std::pair< _T1, _T2 >::pair (
            pair< _T1, _T2 > && )  [constexpr], [default]
```

Move constructor.

pair() [3/5]

```
template<typename _T1, typename _T2>
template<typename _U1 = _T1, typename _U2 = _T2, typename enable_if< __and_< __is_implicitly_↩
default_constructible< _U1 >, __is_implicitly_default_constructible< _U2 > > ::value, bool >↩
::type = true>
std::pair< _T1, _T2 >::pair ()  [inline], [constexpr]
```

The default constructor creates `first` and `second` using their respective default constructors.

pair() [4/5]

```
template<typename _T1, typename _T2>
template<typename _U1 = _T1, typename _U2 = _T2, typename enable_if< _PCCP::template _Constructible↩
Pair< _U1, _U2 >() &&_PCCP::template _ImplicitlyConvertiblePair< _U1, _U2 >(), bool >::type =
true>
std::pair< _T1, _T2 >::pair (
            const _T1 & __a,
            const _T2 & __b)  [inline], [constexpr]
```

Construct from two const lvalues, allowing implicit conversions.

pair() [5/5]

```
template<typename _T1, typename _T2>
template<typename _U1 = _T1, typename _U2 = _T2, typename enable_if< _PCCP::template _Constructible↩
Pair< _U1, _U2 >() &&!_PCCP::template _ImplicitlyConvertiblePair< _U1, _U2 >(), bool >::type =
false>
std::pair< _T1, _T2 >::pair (
            const _T1 & __a,
            const _T2 & __b)  [inline], [explicit], [constexpr]
```

Construct from two const lvalues, disallowing implicit conversions.

5.752.4 Member Function Documentation

swap()

```
template<typename _T1, typename _T2>
void std::pair< _T1, _T2 >::swap (
            pair< _T1, _T2 > & __p)  [inline], [constexpr], [noexcept]
```

Swap the first members and then the second members.

5.752.5 Member Data Documentation

first

```
template<typename _T1, typename _T2>
_T1 std::pair< _T1, _T2 >::first
```

The first member.

second

```
template<typename _T1, typename _T2>
_T2 std::pair< _T1, _T2 >::second
```

The second member.

The documentation for this struct was generated from the following files:

- bits/stl_iterator.h
- stl_pair.h
- tuple

5.753 std::pair< _T1, _T2 > Struct Template Reference

```
#include <utility>
```

Public Types

- typedef _T1 first_type
- typedef _T2 second_type

Public Member Functions

- template<typename _U1 = _T1, typename _U2 = _T2, typename enable_if< __and_< __is_implicitly_default_constructible< _U1 >, __↩ is_implicitly_default_constructible< _U2 > > ::value, bool >::type = true>
  constexpr pair ()
- template<typename _U2, __enable_if_t< __and_< is_pointer< _T1 >, __not_< is_reference< _U2 > >, is_constructible< _T2, _U2 >, __not_< is_constructible< _T2, const _U2 & > >, is_convertible< _U2, _T2 > >::value, bool > = true>
  constexpr **pair** (__zero_as_null_pointer_constant, _U2 &&__y,...)
- template<typename _U2, __enable_if_t< __and_< is_pointer< _T1 >, __not_< is_reference< _U2 > >, is_constructible< _T2, _U2 >, __not_< is_constructible< _T2, const _U2 & > >, __not_< is_convertible< _U2, _T2 > > >::value, bool > = false>
  constexpr **pair** (__zero_as_null_pointer_constant, _U2 &&__y,...)
- template<typename _U1, __enable_if_t< __and_< __not_< is_reference< _U1 > >, is_pointer< _T2 >, is_constructible< _T1, _U1 >, __not_< is_constructible< _T1, const _U1 & > >, is_convertible< _U1, _T1 > >::value, bool > = true>
  constexpr **pair** (_U1 &&__x, __zero_as_null_pointer_constant,...)
- template<typename _U1, __enable_if_t< __and_< __not_< is_reference< _U1 > >, is_pointer< _T2 >, is_constructible< _T1, _U1 >, __not_< is_constructible< _T1, const _U1 & > >, __not_< is_convertible< _U1, _T1 > > >::value, bool > = false>
  constexpr **pair** (_U1 &&__x, __zero_as_null_pointer_constant,...)
- template<typename _U1, typename _U2, typename enable_if< _PCCP::template _MoveConstructiblePair< _U1, _U2 >() &&_PCCP↩ ::template _ImplicitlyMoveConvertiblePair< _U1, _U2 >(), bool >::type = true>
  constexpr **pair** (_U1 &&__x, _U2 &&__y)
- template<typename _U1, typename _U2, typename enable_if< _PCCP::template _MoveConstructiblePair< _U1, _U2 >() &&!_PCCP↩ ::template _ImplicitlyMoveConvertiblePair< _U1, _U2 >(), bool >::type = false>
  constexpr **pair** (_U1 &&__x, _U2 &&__y)
- template<typename _U1 = _T1, typename _U2 = _T2, typename enable_if< _PCCP::template _ConstructiblePair< _U1, _U2 >() &&_↩ PCCP::template _ImplicitlyConvertiblePair< _U1, _U2 >(), bool >::type = true>
  constexpr pair (const _T1 &__a, const _T2 &__b)
- template<typename _U1 = _T1, typename _U2 = _T2, typename enable_if< _PCCP::template _ConstructiblePair< _U1, _U2 >() &&!_↩ PCCP::template _ImplicitlyConvertiblePair< _U1, _U2 >(), bool >::type = false>
  constexpr pair (const _T1 &__a, const _T2 &__b)
- constexpr pair (const pair &)=default
- template<typename _U1, typename _U2, typename enable_if< _PCCFP< _U1, _U2 >::template _ConstructiblePair< _U1, _U2 >() &&↩ _PCCFP< _U1, _U2 >::template _ImplicitlyConvertiblePair< _U1, _U2 >(), bool >::type = true>
  constexpr **pair** (const pair< _U1, _U2 > &__p)
- template<typename _U1, typename _U2, typename enable_if< _PCCFP< _U1, _U2 >::template _ConstructiblePair< _U1, _U2 >() &&!↩ _PCCFP< _U1, _U2 >::template _ImplicitlyConvertiblePair< _U1, _U2 >(), bool >::type = false>
  constexpr **pair** (const pair< _U1, _U2 > &__p)
- constexpr pair (pair &&)=default
- template<typename _U1, typename _U2, typename enable_if< _PCCFP< _U1, _U2 >::template _MoveConstructiblePair< _U1, _U2 >() && _PCCFP< _U1, _U2 >::template _ImplicitlyMoveConvertiblePair< _U1, _U2 >(), bool >::type = true>
  constexpr **pair** (pair< _U1, _U2 > &&__p)
- template<typename _U1, typename _U2, typename enable_if< _PCCFP< _U1, _U2 >::template _MoveConstructiblePair< _U1, _U2 >() &&!_PCCFP< _U1, _U2 >::template _ImplicitlyMoveConvertiblePair< _U1, _U2 >(), bool >::type = false>
  constexpr **pair** (pair< _U1, _U2 > &&__p)
- template<typename... _Args1, typename... _Args2>
  constexpr pair (piecewise_construct_t, tuple< _Args1... >, tuple< _Args2... >)
- pair & **operator=** (__conditional_t< __and_< is_copy_assignable< _T1 >, is_copy_assignable< _T2 > >↩ ::value, const pair &, const __nonesuch & > __p)

- pair & **operator=** (__conditional_t< __and_< is_move_assignable< _T1 >, is_move_assignable< _T2 > >::value, pair &&, __nonesuch && > __p) noexcept(__and_< is_nothrow_move_assignable< _T1 >, is_nothrow_move_assignable< _T2 > >::value)
- template<typename _U1, typename _U2>
  enable_if< __and_< is_assignable< _T1 &, const_U1 & >, is_assignable< _T2 &, const_U2 & > >::value, pair & >::type **operator=** (const pair< _U1, _U2 > &__p)
- template<typename _U1, typename _U2>
  enable_if< __and_< is_assignable< _T1 &, _U1 && >, is_assignable< _T2 &, _U2 && > >::value, pair & >::type **operator=** (pair< _U1, _U2 > &&__p)
- constexpr void swap (pair &__p) noexcept(__and_< __is_nothrow_swappable< _T1 >, __is_nothrow_↩ swappable< _T2 > >::value)

Public Attributes

- _T1 first
- _T2 second

Related Symbols

(Note that these are not member symbols.)

- template<typename _T1, typename _T2>
  pair (_T1, _T2) -> pair< _T1, _T2 >
- template<typename _T1, typename _T2, typename _U1, typename _U2>
  constexpr bool operator== (const pair< _T1, _T2 > &__x, const pair< _U1, _U2 > &__y)
- template<typename _T1, typename _T2, typename _U1, typename _U2>
  constexpr common_comparison_category_t< __detail::__synth3way_t< _T1, _U1 >, __detail::__synth3way_t< _T2, _U2 > > operator<=> (const pair< _T1, _T2 > &__x, const pair< _U1, _U2 > &__y)
- template<typename _T1, typename _T2>
  constexpr enable_if< __and_< __is_swappable< _T1 >, __is_swappable< _T2 > >::value >::type swap (pair< _T1, _T2 > &__x, pair< _T1, _T2 > &__y) noexcept(noexcept(__x.swap(__y)))

5.753.1 Detailed Description

template<typename _T1, typename _T2>
struct std::pair< _T1, _T2 >

Struct holding two objects of arbitrary type.

Template Parameters

_T1	Type of first object.
_T2	Type of second object.

https://gcc.gnu.org/onlinedocs/libstdc++/manual/utilities.html

5.753.2 Member Typedef Documentation

first_type

```
template<typename _T1, typename _T2>
typedef _T1 std::pair< _T1, _T2 >::first_type
```

The type of the `first` member.

second_type

```
template<typename _T1, typename _T2>
typedef _T2 std::pair< _T1, _T2 >::second_type
```

The type of the `second` member.

5.753.3 Constructor & Destructor Documentation

pair() [1/5]

```
template<typename _T1, typename _T2>
std::pair< _T1, _T2 >::pair (
            const pair< _T1, _T2 > & )  [constexpr], [default]
```

Copy constructor.

pair() [2/5]

```
template<typename _T1, typename _T2>
std::pair< _T1, _T2 >::pair (
            pair< _T1, _T2 > && )  [constexpr], [default]
```

Move constructor.

pair() [3/5]

```
template<typename _T1, typename _T2>
template<typename _U1 = _T1, typename _U2 = _T2, typename enable_if< __and_< __is_implicitly_↩
default_constructible< _U1 >, __is_implicitly_default_constructible< _U2 > > ::value, bool >↩
::type = true>
std::pair< _T1, _T2 >::pair ()  [inline], [constexpr]
```

The default constructor creates `first` and `second` using their respective default constructors.

pair() [4/5]

```
template<typename _T1, typename _T2>
template<typename _U1 = _T1, typename _U2 = _T2, typename enable_if< _PCCP::template _Constructible↩
Pair< _U1, _U2 >() &&_PCCP::template _ImplicitlyConvertiblePair< _U1, _U2 >(), bool >::type =
true>
std::pair< _T1, _T2 >::pair (
            const _T1 & __a,
            const _T2 & __b)  [inline], [constexpr]
```

Construct from two const lvalues, allowing implicit conversions.

pair() [5/5]

```
template<typename _T1, typename _T2>
template<typename _U1 = _T1, typename _U2 = _T2, typename enable_if< _PCCP::template _Constructible↩
Pair< _U1, _U2 >() &&!_PCCP::template _ImplicitlyConvertiblePair< _U1, _U2 >(), bool >::type =
false>
std::pair< _T1, _T2 >::pair (
            const _T1 & __a,
            const _T2 & __b)  [inline], [explicit], [constexpr]
```

Construct from two const lvalues, disallowing implicit conversions.

5.753.4 Member Function Documentation

swap()

```
template<typename _T1, typename _T2>
void std::pair< _T1, _T2 >::swap (
            pair< _T1, _T2 > & __p)  [inline], [constexpr], [noexcept]
```

Swap the first members and then the second members.
Referenced by std::pair< const __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator > >::swap(), std::sub_match< const char * and std::swap().

5.753.5 Member Data Documentation

first

```
template<typename _T1, typename _T2>
_T1 std::pair< _T1, _T2 >::first
```

The first member.
Referenced by __gnu_parallel::__find_template(), __gnu_parallel::__find_template(), std::__sample(), std::set< _Key, _Cmp, polymorphic std::pair< const __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator > >::operator<=>(), std::pair< const __iter_key_t< _Input and std::shuffle().

second

```
template<typename _T1, typename _T2>
_T2 std::pair< _T1, _T2 >::second
```

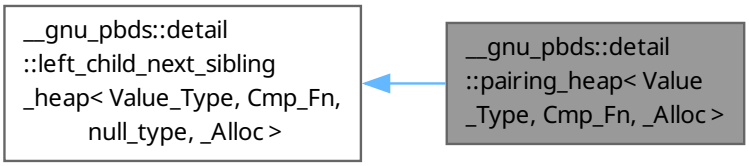
The second member.
Referenced by std::__sample(), std::set< _Key, _Cmp, polymorphic_allocator< _Key > >::insert(), std::pair< const __iter_key_t< _Input std::pair< const __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator > >::operator==(), and std::shuffle().
The documentation for this struct was generated from the following files:

- bits/stl_iterator.h
- stl_pair.h
- tuple

5.754 __gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc > Class Template Reference

```
#include <pairing_heap_.hpp>
```

Inheritance diagram for __gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc >:



Public Types

- typedef _Alloc **allocator_type**
- typedef Cmp_Fn **cmp_fn**
- typedef base_type::const_iterator **const_iterator**
- typedef __rebind_a::const_pointer **const_pointer**
- typedef __rebind_a::const_reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef base_type::iterator **iterator**
- typedef left_child_next_sibling_heap_node_< Value_Type, null_type, _Alloc > **node**
- typedef base_type::point_const_iterator **point_const_iterator**
- typedef base_type::point_iterator **point_iterator**
- typedef __rebind_a::pointer **pointer**
- typedef __rebind_a::reference **reference**
- typedef _Alloc::size_type **size_type**
- typedef Value_Type **value_type**

Public Member Functions

- **pairing_heap** (const Cmp_Fn &)
- **pairing_heap** (const pairing_heap &)
- iterator **begin** ()
- iterator **begin** ()
- const_iterator **begin** () const
- const_iterator **begin** () const
- void **clear** ()
- void **clear** ()
- bool **empty** () const
- bool **empty** () const
- iterator **end** ()
- iterator **end** ()
- const_iterator **end** () const
- const_iterator **end** () const
- void **erase** (point_iterator)
- template<typename Pred>
  size_type **erase_if** (Pred)
- Cmp_Fn & **get_cmp_fn** ()
- Cmp_Fn & **get_cmp_fn** ()
- const Cmp_Fn & **get_cmp_fn** () const
- const Cmp_Fn & **get_cmp_fn** () const
- void **join** (pairing_heap &)
- size_type **max_size** () const
- size_type **max_size** () const
- void **modify** (point_iterator, const_reference)
- void **pop** ()
- point_iterator **push** (const_reference)
- size_type **size** () const
- size_type **size** () const
- template<typename Pred>
  void **split** (Pred, pairing_heap &)
- void **swap** (left_child_next_sibling_heap< Value_Type, Cmp_Fn, null_type, _Alloc > &)
- void **swap** (left_child_next_sibling_heap< Value_Type, Cmp_Fn, null_type, _Alloc > &)
- void **swap** (pairing_heap &)
- const_reference **top** () const

Protected Types

- typedef alloc_traits::allocator_type **node_allocator**
- typedef alloc_traits::const_pointer **node_const_pointer**
- typedef null_type **node_metadata**
- typedef std::pair< node_pointer, node_pointer > **node_pointer_pair**

Protected Member Functions

- void **actual_erase_node** (node_pointer)
- void **actual_erase_node** (node_pointer)
- void **bubble_to_top** (node_pointer)
- void **bubble_to_top** (node_pointer)
- void **clear_imp** (node_pointer)
- void **clear_imp** (node_pointer)
- template<typename It>
  void **copy_from_range** (It, It)
- node_pointer **get_new_node_for_insert** (const_reference)
- node_pointer **get_new_node_for_insert** (const_reference)
- node_pointer **prune** (Pred)
- node_pointer **prune** (Pred)
- void **swap_with_parent** (node_pointer, node_pointer)
- void **swap_with_parent** (node_pointer, node_pointer)
- void **to_linked_list** ()
- void **to_linked_list** ()
- void **value_swap** (left_child_next_sibling_heap &)
- void **value_swap** (left_child_next_sibling_heap &)

Static Protected Member Functions

- static void **make_child_of** (node_pointer, node_pointer)
- static void **make_child_of** (node_pointer, node_pointer)
- static node_pointer **parent** (node_pointer)
- static node_pointer **parent** (node_pointer)

Protected Attributes

- node_pointer **m_p_root**
- node_pointer **m_p_root**
- size_type **m_size**
- size_type **m_size**

5.754.1 Detailed Description

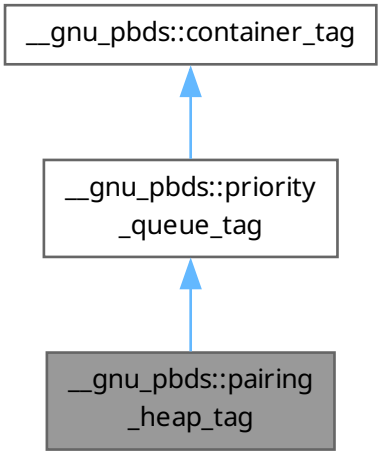
template<typename Value_Type, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc >

Pairing heap.
The documentation for this class was generated from the following file:

- pairing_heap_.hpp

5.755 __gnu_pbds::pairing_heap_tag Struct Reference

`#include <tag_and_trait.hpp>`
Inheritance diagram for __gnu_pbds::pairing_heap_tag:



5.755.1 Detailed Description

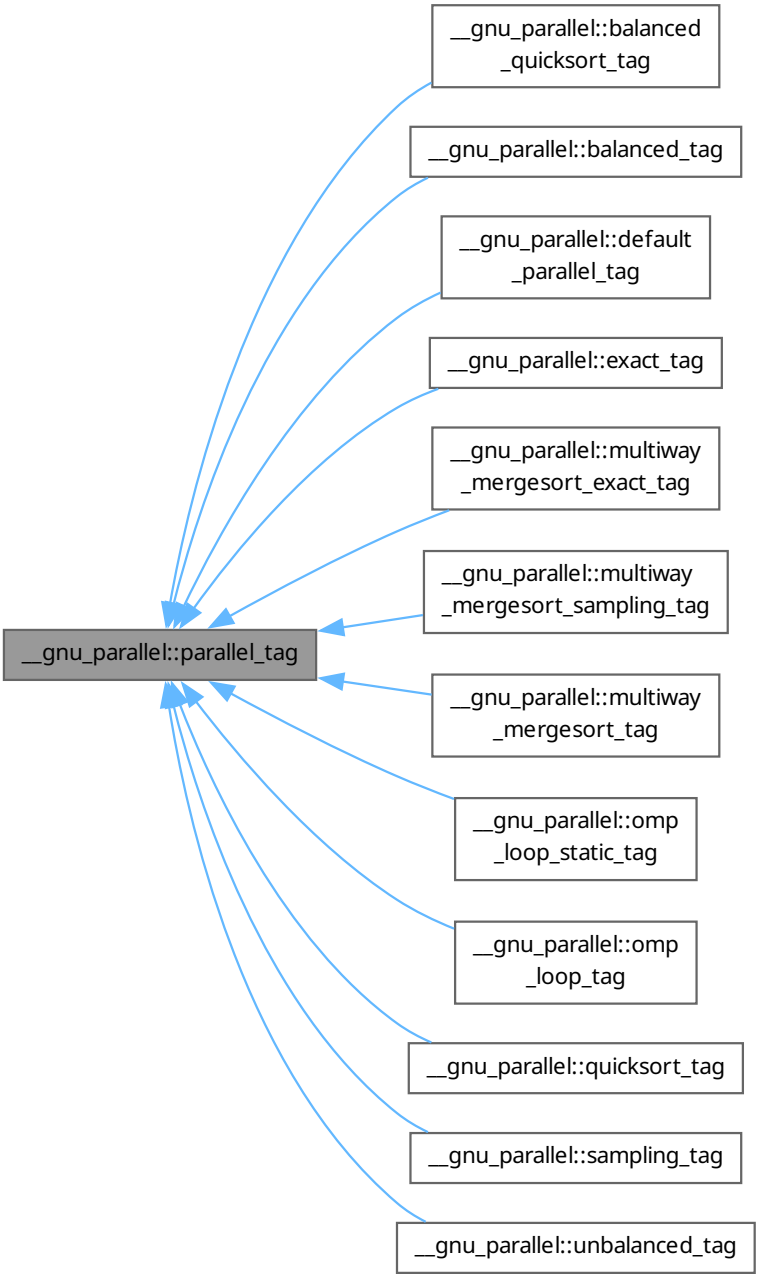
Pairing-heap.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

5.756 __gnu_parallel::parallel_tag Struct Reference

`#include <tags.h>`

Inheritance diagram for __gnu_parallel::parallel_tag:



Public Member Functions

- parallel_tag ()
- parallel_tag (_ThreadIndex __num_threads)
- _ThreadIndex __get_num_threads ()

- void set_num_threads (_ThreadIndex __num_threads)

5.756.1 Detailed Description

Recommends parallel execution at compile time, optionally using a user-specified number of threads.

5.756.2 Constructor & Destructor Documentation

parallel_tag() [1/2]

```
__gnu_parallel::parallel_tag::parallel_tag ()  [inline]
```

Default constructor. Use default number of threads.

parallel_tag() [2/2]

```
__gnu_parallel::parallel_tag::parallel_tag (
            _ThreadIndex __num_threads)  [inline]
```

Default constructor. Recommend number of threads to use.

Parameters

__num_threads	Desired number of threads.

5.756.3 Member Function Documentation

__get_num_threads()

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ()  [inline]
```

Find out desired number of threads.

Returns

Desired number of threads.

Referenced by __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), and __gnu_parallel::__parallel_sort(

set_num_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
            _ThreadIndex __num_threads)  [inline]
```

Set the desired number of threads.

Parameters

__num_threads	Desired number of threads.

The documentation for this struct was generated from the following file:

- tags.h

5.757 std::bernoulli_distribution::param_type Struct Reference

```
#include <random.h>
```

Public Types

- typedef bernoulli_distribution **distribution_type**

Public Member Functions

- **param_type** (double __p)
- double **p** () const

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)

5.757.1 Detailed Description

Parameter type.
The documentation for this struct was generated from the following file:

- random.h

5.758 std::binomial_distribution< _IntType >::param_type Struct Reference

`#include <random.h>`

Public Types

- typedef binomial_distribution< _IntType > **distribution_type**

Public Member Functions

- **param_type** (_IntType __t, double __p=0.5)
- double **p** () const
- _IntType **t** () const

Friends

- class **binomial_distribution< _IntType >**
- bool **operator==** (const param_type &__p1, const param_type &__p2)

5.758.1 Detailed Description

template<typename _IntType = int>
struct std::binomial_distribution< _IntType >::param_type

Parameter type.
The documentation for this struct was generated from the following files:

- random.h
- bits/random.tcc

5.759 std::cauchy_distribution< _RealType >::param_type Struct Reference

`#include <random.h>`

Public Types

- typedef cauchy_distribution< _RealType > **distribution_type**

Public Member Functions

- **param_type** (_RealType __a, _RealType __b=_RealType(1))
- _RealType **a** () const
- _RealType **b** () const

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)

5.759.1 Detailed Description

template<typename _RealType = double>
struct std::cauchy_distribution< _RealType >::param_type

Parameter type.
The documentation for this struct was generated from the following file:

- random.h

5.760 std::chi_squared_distribution< _RealType >::param_type Struct Reference

`#include <random.h>`

Public Types

- typedef chi_squared_distribution< _RealType > **distribution_type**

Public Member Functions

- **param_type** (_RealType __n)
- _RealType **n** () const

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)

5.760.1 Detailed Description

template<typename _RealType = double>
struct std::chi_squared_distribution< _RealType >::param_type

Parameter type.
The documentation for this struct was generated from the following file:

- random.h

5.761 std::discrete_distribution< _IntType >::param_type Struct Reference

`#include <random.h>`

Public Types

- typedef discrete_distribution< _IntType > **distribution_type**

Public Member Functions

- template<typename _InputIterator>
  param_type (_InputIterator __wbegin, _InputIterator __wend)
- **param_type** (const param_type &)=default
- **param_type** (initializer_list< double > __wil)
- template<typename _Func>
  param_type (size_t __nw, double __xmin, double __xmax, _Func __fw)
- param_type & **operator=** (const param_type &)=default
- std::vector< double > **probabilities** () const

Friends

- class **discrete_distribution**< **_IntType** >
- bool **operator==** (const param_type &__p1, const param_type &__p2)

5.761.1 Detailed Description

template<typename _IntType = int>
struct std::discrete_distribution< _IntType >::param_type

Parameter type.
The documentation for this struct was generated from the following files:

- random.h
- bits/random.tcc

5.762 std::exponential_distribution< _RealType >::param_type Struct Reference

`#include <random.h>`

Public Types

- typedef exponential_distribution< _RealType > **distribution_type**

Public Member Functions

- **param_type** (_RealType __lambda)
- _RealType **lambda** () const

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)

5.762.1 Detailed Description

template<typename _RealType = double>
struct std::exponential_distribution< _RealType >::param_type

Parameter type.
The documentation for this struct was generated from the following file:

- random.h

5.763 std::extreme_value_distribution< _RealType >::param_type Struct Reference

`#include <random.h>`

Public Types

- typedef extreme_value_distribution< _RealType > **distribution_type**

Public Member Functions

- **param_type** (_RealType __a, _RealType __b=_RealType(1.0))
- _RealType **a** () const
- _RealType **b** () const

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)

5.763.1 Detailed Description

template<typename _RealType = double>
struct std::extreme_value_distribution< _RealType >::param_type

Parameter type.
The documentation for this struct was generated from the following file:

- random.h

5.764 std::fisher_f_distribution< _RealType >::param_type Struct Reference

`#include <random.h>`

Public Types

- typedef fisher_f_distribution< _RealType > **distribution_type**

Public Member Functions

- **param_type** (_RealType __m, _RealType __n=_RealType(1))
- _RealType **m** () const
- _RealType **n** () const

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)

5.764.1 Detailed Description

template<typename _RealType = double>
struct std::fisher_f_distribution< _RealType >::param_type

Parameter type.
The documentation for this struct was generated from the following file:

- random.h

5.765 std::gamma_distribution< _RealType >::param_type Struct Reference

`#include <random.h>`

Public Types

- typedef gamma_distribution< _RealType > **distribution_type**

Public Member Functions

- **param_type** (_RealType __alpha_val, _RealType __beta_val=_RealType(1))
- _RealType **alpha** () const
- _RealType **beta** () const

Friends

- class **gamma_distribution< _RealType >**
- bool **operator==** (const param_type &__p1, const param_type &__p2)

5.765.1 Detailed Description

template<typename _RealType = double>
struct std::gamma_distribution< _RealType >::param_type

Parameter type.
The documentation for this struct was generated from the following files:

- random.h
- bits/random.tcc

5.766 std::geometric_distribution< _IntType >::param_type Struct Reference

`#include <random.h>`

Public Types

- typedef geometric_distribution< _IntType > **distribution_type**

Public Member Functions

- **param_type** (double __p)
- double **p** () const

Friends

- class **geometric_distribution< _IntType >**
- bool **operator==** (const param_type &__p1, const param_type &__p2)

5.766.1 Detailed Description

template<typename _IntType = int>
struct std::geometric_distribution< _IntType >::param_type

Parameter type.
The documentation for this struct was generated from the following file:

- random.h

5.767 std::lognormal_distribution< _RealType >::param_type Struct Reference

`#include <random.h>`

Public Types

- typedef lognormal_distribution< _RealType > **distribution_type**

Public Member Functions

- **param_type** (_RealType __m, _RealType __s=_RealType(1))
- _RealType **m** () const
- _RealType **s** () const

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)

5.767.1 Detailed Description

template<typename _RealType = double>
struct std::lognormal_distribution< _RealType >::param_type

Parameter type.
The documentation for this struct was generated from the following file:

- random.h

5.768 std::negative_binomial_distribution< _IntType >::param_type Struct Reference

`#include <random.h>`

Public Types

- typedef negative_binomial_distribution< _IntType > **distribution_type**

Public Member Functions

- **param_type** (_IntType __k, double __p=0.5)
- _IntType **k** () const
- double **p** () const

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)

5.768.1 Detailed Description

template<typename _IntType = int>
struct std::negative_binomial_distribution< _IntType >::param_type

Parameter type.
The documentation for this struct was generated from the following file:

- random.h

5.769 std::normal_distribution< _RealType >::param_type Struct Reference

`#include <random.h>`

Public Types

- typedef normal_distribution< _RealType > **distribution_type**

Public Member Functions

- **param_type** (_RealType __mean, _RealType __stddev=_RealType(1))
- _RealType **mean** () const
- _RealType **stddev** () const

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)

5.769.1 Detailed Description

template<typename _RealType = double>
struct std::normal_distribution< _RealType >::param_type

Parameter type.
The documentation for this struct was generated from the following file:

- random.h

5.770 std::piecewise_constant_distribution< _RealType >::param_type Struct Reference

`#include <random.h>`

Public Types

- typedef piecewise_constant_distribution< _RealType > **distribution_type**

Public Member Functions

- template<typename _InputIteratorB, typename _InputIteratorW>
  param_type (_InputIteratorB __bfirst, _InputIteratorB __bend, _InputIteratorW __wbegin)
- **param_type** (const param_type &)=default
- template<typename _Func>
  param_type (initializer_list< _RealType > __bi, _Func __fw)
- template<typename _Func>
  param_type (size_t __nw, _RealType __xmin, _RealType __xmax, _Func __fw)
- std::vector< double > **densities** () const
- std::vector< _RealType > **intervals** () const
- param_type & **operator=** (const param_type &)=default

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)
- class **piecewise_constant_distribution< _RealType >**

5.770.1 Detailed Description

template<typename _RealType = double>
struct std::piecewise_constant_distribution< _RealType >::param_type

Parameter type.
The documentation for this struct was generated from the following files:

- random.h
- bits/random.tcc

5.771 std::piecewise_linear_distribution< _RealType >::param_type Struct Reference

`#include <random.h>`

Public Types

- typedef piecewise_linear_distribution< _RealType > **distribution_type**

Public Member Functions

- template<typename _InputIteratorB, typename _InputIteratorW>
  param_type (_InputIteratorB __bfirst, _InputIteratorB __bend, _InputIteratorW __wbegin)
- **param_type** (const param_type &)=default
- template<typename _Func>
  param_type (initializer_list< _RealType > __bl, _Func __fw)
- template<typename _Func>
  param_type (size_t __nw, _RealType __xmin, _RealType __xmax, _Func __fw)
- std::vector< double > **densities** () const
- std::vector< _RealType > **intervals** () const
- param_type & **operator=** (const param_type &)=default

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)
- class **piecewise_linear_distribution< _RealType >**

5.771.1 Detailed Description

template<typename _RealType = double>
struct std::piecewise_linear_distribution< _RealType >::param_type

Parameter type.
The documentation for this struct was generated from the following files:

- random.h
- bits/random.tcc

5.772 std::poisson_distribution< _IntType >::param_type Struct Reference

`#include <random.h>`

Public Types

- typedef poisson_distribution< _IntType > **distribution_type**

Public Member Functions

- **param_type** (double __mean)
- double **mean** () const

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)
- class **poisson_distribution**< **_IntType** >

5.772.1 Detailed Description

template<**typename _IntType = int**>
struct std::poisson_distribution< **_IntType** >**::param_type**

Parameter type.
The documentation for this struct was generated from the following files:

- random.h
- bits/random.tcc

5.773 std::student_t_distribution< _RealType >::param_type Struct Reference

`#include <random.h>`

Public Types

- typedef student_t_distribution< _RealType > **distribution_type**

Public Member Functions

- **param_type** (_RealType __n)
- _RealType **n** () const

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)

5.773.1 Detailed Description

template<**typename _RealType = double**>
struct std::student_t_distribution< **_RealType** >**::param_type**

Parameter type.
The documentation for this struct was generated from the following file:

- random.h

5.774 std::uniform_int_distribution< _IntType >::param_type Struct Reference

`#include <uniform_int_dist.h>`

Public Types

- typedef uniform_int_distribution< _IntType > **distribution_type**

Public Member Functions

- **param_type** (_IntType __a, _IntType __b=__gnu_cxx::__int_traits< _IntType >::__max)
- result_type **a** () const
- result_type **b** () const

Friends

- bool **operator!=** (const param_type &__p1, const param_type &__p2)
- bool **operator==** (const param_type &__p1, const param_type &__p2)

5.774.1 Detailed Description

template<typename _IntType = int>
struct std::uniform_int_distribution< _IntType >::param_type

Parameter type.
The documentation for this struct was generated from the following file:

- uniform_int_dist.h

5.775 std::uniform_real_distribution< _RealType >::param_type Struct Reference

`#include <random.h>`

Public Types

- typedef uniform_real_distribution< _RealType > **distribution_type**

Public Member Functions

- **param_type** (_RealType __a, _RealType __b=_RealType(1))
- result_type **a** () const
- result_type **b** () const

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)

5.775.1 Detailed Description

template<typename _RealType = double>
struct std::uniform_real_distribution< _RealType >::param_type

Parameter type.
The documentation for this struct was generated from the following file:

- random.h

5.776 std::weibull_distribution< _RealType >::param_type Struct Reference

`#include <random.h>`

Public Types

- typedef weibull_distribution< _RealType > **distribution_type**

Public Member Functions

- **param_type** (_RealType __a, _RealType __b=_RealType(1.0))
- _RealType **a** () const
- _RealType **b** () const

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)

5.776.1 Detailed Description

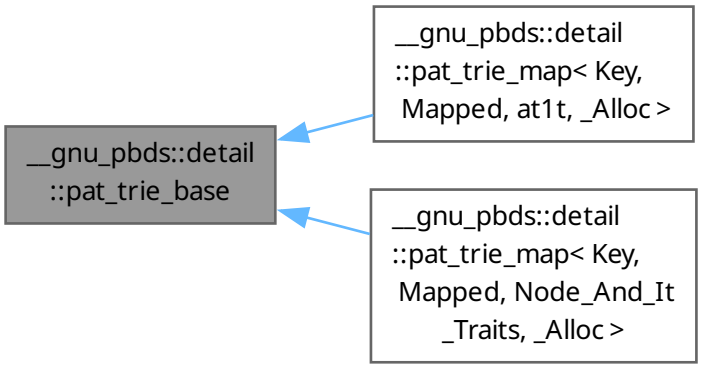
template<typename _RealType = double>
struct std::weibull_distribution< _RealType >::param_type

Parameter type.
The documentation for this struct was generated from the following file:

- random.h

5.777 __gnu_pbds::detail::pat_trie_base Struct Reference

`#include <pat_trie_base.hpp>`
Inheritance diagram for __gnu_pbds::detail::pat_trie_base:



Classes

- class _CIter
- struct _Head
- struct _Inode
- class _Iter
- struct _Leaf
- struct _Metadata
- struct _Metadata< null_type, _Alloc >
- struct _Node_base

- class _Node_citer
- class _Node_iter

Public Types

- enum node_type { **i_node** , **leaf_node** , **head_node** }

5.777.1 Detailed Description

Base type for PATRICIA trees.

5.777.2 Member Enumeration Documentation

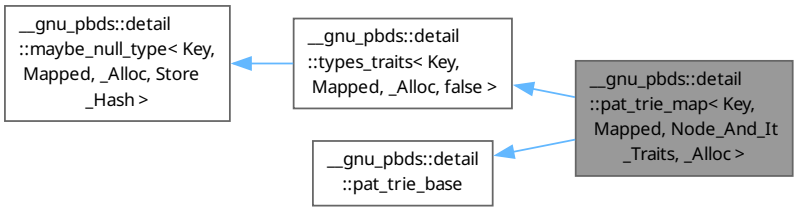
node_type

`enum __gnu_pbds::detail::pat_trie_base::node_type`
Three types of nodes.
i_node is used by _Inode, leaf_node by _Leaf, and head_node by _Head.
The documentation for this struct was generated from the following file:

- pat_trie_base.hpp

5.778 __gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc > Class Template Reference

`#include <pat_trie_.hpp>`
Inheritance diagram for __gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >:



Public Types

- typedef traits_type::access_traits **access_traits**
- typedef _Alloc **allocator_type**
- typedef std::pair< size_type, size_type > **comp_hash**
- typedef point_const_iterator **const_iterator**
- typedef traits_base::const_pointer **const_pointer**
- typedef traits_base::const_reference **const_reference**
- typedef traits_type::const_reverse_iterator **const_reverse_iterator**
- typedef pat_trie_tag **container_category**
- typedef _Alloc::difference_type **difference_type**
- typedef point_iterator **iterator**
- typedef traits_base::key_const_pointer **key_const_pointer**
- typedef traits_base::key_const_reference **key_const_reference**

- typedef traits_base::key_pointer **key_pointer**
- typedef traits_base::key_reference **key_reference**
- typedef traits_base::key_type **key_type**
- typedef traits_base::mapped_const_pointer **mapped_const_pointer**
- typedef traits_base::mapped_const_reference **mapped_const_reference**
- typedef traits_base::mapped_pointer **mapped_pointer**
- typedef traits_base::mapped_reference **mapped_reference**
- typedef traits_base::mapped_type **mapped_type**
- typedef __nothrowcopy::indicator **no_throw_indicator**
- typedef traits_type::node_const_iterator **node_const_iterator**
- typedef traits_type::node_iterator **node_iterator**
- enum node_type { **i_node** , **leaf_node** , **head_node** }
- typedef traits_type::node_update **node_update**
- typedef traits_type::const_iterator **point_const_iterator**
- typedef traits_type::iterator **point_iterator**
- typedef traits_base::pointer **pointer**
- typedef traits_base::reference **reference**
- typedef traits_type::reverse_iterator **reverse_iterator**
- typedef _Alloc::size_type **size_type**
- typedef integral_constant< int, Store_Hash > **store_extra**
- typedef stored_data< value_type, size_type, Store_Hash > **stored_data_type**
- typedef traits_base::value_type **value_type**

Public Member Functions

- **pat_trie_map** (const access_traits &)
- **pat_trie_map** (const pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc > &)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- const_iterator **erase** (const_iterator)
- const_reverse_iterator **erase** (const_reverse_iterator)
- iterator **erase** (iterator)
- bool **erase** (key_const_reference)
- reverse_iterator **erase** (reverse_iterator)
- template<typename Pred>
  size_type **erase_if** (Pred)
- point_iterator **find** (key_const_reference)
- point_const_iterator **find** (key_const_reference) const
- access_traits & **get_access_traits** ()
- const access_traits & **get_access_traits** () const
- node_update & **get_node_update** ()
- const node_update & **get_node_update** () const
- std::pair< point_iterator, bool > **insert** (const_reference)
- void **join** (pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc > &)
- point_iterator **lower_bound** (key_const_reference)
- point_const_iterator **lower_bound** (key_const_reference) const
- size_type **max_size** () const

- node_iterator node_begin ()
- node_const_iterator node_begin () const
- node_iterator node_end ()
- node_const_iterator node_end () const
- mapped_reference **operator[ ]** (key_const_reference r_key)
- reverse_iterator **rbegin** ()
- const_reverse_iterator **rbegin** () const
- reverse_iterator **rend** ()
- const_reverse_iterator **rend** () const
- size_type **size** () const
- void **split** (key_const_reference, pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc > &)
- void **swap** (pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc > &)
- point_iterator **upper_bound** (key_const_reference)
- point_const_iterator **upper_bound** (key_const_reference) const

Public Attributes

- no_throw_indicator **m_no_throw_copies_indicator**
- no_throw_indicator **m_no_throw_copies_indicator**
- store_extra **m_store_extra_indicator**
- store_extra **m_store_extra_indicator**

Protected Member Functions

- template<typename It>
  void **copy_from_range** (It, It)
- node_pointer **recursive_copy_node** (node_const_pointer)
- void **value_swap** (pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc > &)

5.778.1 Detailed Description

template<typename Key, typename Mapped, typename Node_And_It_Traits, typename _Alloc>
class __gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >

PATRICIA trie.
This implementation loosely borrows ideas from: 1) Fast Mergeable Integer Maps, Okasaki, Gill 1998 2) Ptset: Sets of integers implemented as Patricia trees, Jean-Christophe Filliatr, 2000

5.778.2 Member Enumeration Documentation

node_type

```
enum __gnu_pbds::detail::pat_trie_base::node_type  [inherited]
```

Three types of nodes.
i_node is used by _Inode, leaf_node by _Leaf, and head_node by _Head.

5.778.3 Member Function Documentation

node_begin() [1/2]

```
template<typename Key, typename Mapped, typename Node_And_It_Traits, typename _Alloc>
pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_iterator __gnu_pbds::detail::pat_trie_map<
Key, Mapped, Node_And_It_Traits, _Alloc >::node_begin ()  [inline]
```

Returns a node_iterator corresponding to the node at the root of the tree.

node_begin() [2/2]

```
template<typename Key, typename Mapped, typename Node_And_It_Traits, typename _Alloc>
pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_const_iterator __gnu_pbds::detail::pat_trie_map<
Key, Mapped, Node_And_It_Traits, _Alloc >::node_begin () const  [inline]
```

Returns a const node_iterator corresponding to the node at the root of the tree.

node_end() [1/2]

```
template<typename Key, typename Mapped, typename Node_And_It_Traits, typename _Alloc>
pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_iterator __gnu_pbds::detail::pat_trie_map<
Key, Mapped, Node_And_It_Traits, _Alloc >::node_end ()  [inline]
```

Returns a node_iterator corresponding to a node just after a leaf of the tree.

node_end() [2/2]

```
template<typename Key, typename Mapped, typename Node_And_It_Traits, typename _Alloc>
pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_const_iterator __gnu_pbds::detail::pat_trie_map<
Key, Mapped, Node_And_It_Traits, _Alloc >::node_end () const  [inline]
```

Returns a const node_iterator corresponding to a node just after a leaf of the tree.

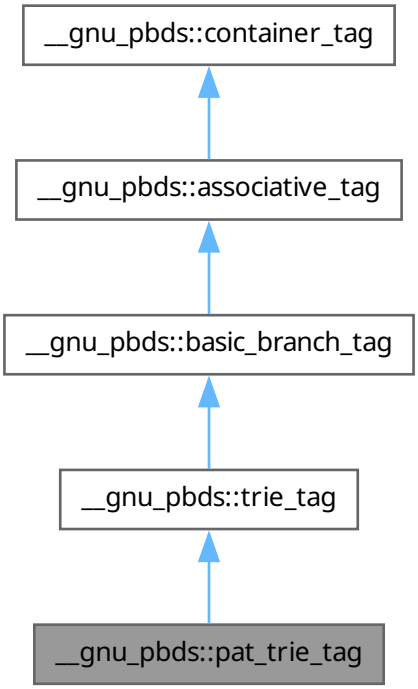
The documentation for this class was generated from the following file:

- pat_trie_.hpp

5.779 __gnu_pbds::pat_trie_tag Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for __gnu_pbds::pat_trie_tag:



5.779.1 Detailed Description

PATRICIA trie.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

5.780 std::experimental::filesystem::v1::path Class Reference

`#include <fs_path.h>`

Classes

- class iterator

Public Types

- typedef iterator **const_iterator**
- typedef std::basic_string< value_type > **string_type**
- typedef char **value_type**

Public Member Functions

- template<typename _InputIterator, typename _Require = __detail::_Path<_InputIterator, _InputIterator>>
  path (_InputIterator __first, _InputIterator __last)
- template<typename _InputIterator, typename _Require = __detail::_Path<_InputIterator, _InputIterator>, typename _Require2 = __detail::__value_type_is_char<_InputIterator>>
  path (_InputIterator __first, _InputIterator __last, const locale &__loc)
- template<typename _Source, typename _Require = __detail::_Path<_Source>>
  path (_Source const &__source)
- template<typename _Source, typename _Require = __detail::_Path<_Source>, typename _Require2 = __detail::__value_type_is_char<_Source>>
  path (_Source const &__source, const locale &__loc)
- **path** (const path &__p)
- **path** (path &&__p) noexcept
- **path** (string_type &&__source)
- template<typename _InputIterator>
  __detail::_Path< _InputIterator, _InputIterator > & **append** (_InputIterator __first, _InputIterator __last)
- template<typename _Source>
  __detail::_Path< _Source > & **append** (_Source const &__source)
- template<typename _InputIterator>
  __detail::_Path< _InputIterator, _InputIterator > & **assign** (_InputIterator __first, _InputIterator __last)
- template<typename _Source>
  __detail::_Path< _Source > & **assign** (_Source const &__source)
- path & **assign** (string_type &&__source)
- iterator **begin** () const noexcept
- const value_type * **c_str** () const noexcept
- void **clear** () noexcept
- int **compare** (const basic_string_view< value_type > __s) const
- int **compare** (const path &__p) const noexcept
- int **compare** (const string_type &__s) const
- int **compare** (const value_type *__s) const
- template<typename _InputIterator>
  __detail::_Path< _InputIterator, _InputIterator > & **concat** (_InputIterator __first, _InputIterator __last)
- template<typename _Source>
  __detail::_Path< _Source > & **concat** (_Source const &__x)
- bool **empty** () const noexcept
- iterator **end** () const noexcept
- path **extension** () const
- path **filename** () const
- std::string **generic_string** () const
- template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>>
  std::basic_string< _CharT, _Traits, _Allocator > **generic_string** (const _Allocator &__a=_Allocator()) const
- std::u16string **generic_u16string** () const
- std::u32string **generic_u32string** () const
- std::string **generic_u8string** () const
- std::wstring **generic_wstring** () const
- bool **has_extension** () const
- bool **has_filename** () const
- bool **has_parent_path** () const
- bool **has_relative_path** () const
- bool **has_root_directory** () const
- bool **has_root_name** () const

- bool **has_root_path** () const
- bool **has_stem** () const
- bool **is_absolute** () const
- bool **is_relative** () const
- path & **make_preferred** ()
- const string_type & **native** () const noexcept
- **operator string_type** () const
- template<typename _CharT>
  __detail::_Path< _CharT *, _CharT * > & **operator+=** (_CharT __x)
- template<typename _Source>
  __detail::_Path< _Source > & **operator+=** (_Source const &__x)
- path & **operator+=** (basic_string_view< value_type > __x)
- path & **operator+=** (const path &__x)
- path & **operator+=** (const string_type &__x)
- path & **operator+=** (const value_type *__x)
- path & **operator+=** (value_type __x)
- template<typename _Source>
  __detail::_Path< _Source > & **operator/=** (_Source const &__source)
- path & **operator/=** (const path &__p)
- template<typename _Source>
  __detail::_Path< _Source > & **operator=** (_Source const &__source)
- path & **operator=** (const path &__p)
- path & **operator=** (path &&__p) noexcept
- path & **operator=** (string_type &&__source)
- path **parent_path** () const
- path **relative_path** () const
- path & **remove_filename** ()
- path & **replace_extension** (const path &__replacement=path())
- path & **replace_filename** (const path &__replacement)
- path **root_directory** () const
- path **root_name** () const
- path **root_path** () const
- path **stem** () const
- std::string **string** () const
- template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>>
  std::basic_string< _CharT, _Traits, _Allocator > **string** (const _Allocator &__a=_Allocator()) const
- void **swap** (path &__rhs) noexcept
- std::u16string **u16string** () const
- std::u32string **u32string** () const
- std::string **u8string** () const
- std::wstring **wstring** () const

Static Public Attributes

- static constexpr value_type **preferred_separator**

5.780.1 Detailed Description

A filesystem path.
The documentation for this class was generated from the following file:

- experimental/bits/fs_path.h

5.781 std::filesystem::path Class Reference

`#include <filesystem>`

Classes

- class iterator

Public Types

- using **const_iterator**
- enum format : unsigned char { **native_format** , **generic_format** , **auto_format** }
- using **string_type**
- using **value_type**

Public Member Functions

- template<typename _InputIterator, typename _Require = __detail::_Path2<_InputIterator>, typename _Req2 = __detail::__value_type_is_char<_InputIterator>>
  path (_InputIterator __first, _InputIterator __last, const locale &__loc, format=auto_format)
- template<typename _InputIterator, typename _Require = __detail::_Path2<_InputIterator>>
  path (_InputIterator __first, _InputIterator __last, format=auto_format)
- template<typename _Source, typename _Require = __detail::_Path<_Source>>
  path (_Source const &__source, format=auto_format)
- template<typename _Source, typename _Require = __detail::_Path<_Source>, typename _Require2 = __detail::__value_type_is_char<_Source>>
  path (_Source const &__src, const locale &__loc, format=auto_format)
- **path** (const path &__p)=default
- **path** (path &&__p) noexcept
- **path** (string_type &&__source, format=auto_format)
- template<typename _InputIterator>
  __detail::_Path2< _InputIterator > & **append** (_InputIterator __first, _InputIterator __last)
- template<typename _Source>
  __detail::_Path< _Source > & **append** (_Source const &__source)
- template<typename _InputIterator>
  __detail::_Path2< _InputIterator > & **assign** (_InputIterator __first, _InputIterator __last)
- template<typename _Source>
  __detail::_Path< _Source > & **assign** (_Source const &__source)
- path & **assign** (string_type &&__source)
- iterator **begin** () const noexcept
- const value_type * **c_str** () const noexcept
- void **clear** () noexcept
- int **compare** (basic_string_view< value_type > __s) const noexcept
- int **compare** (const path &__p) const noexcept
- int **compare** (const string_type &__s) const noexcept
- int **compare** (const value_type *__s) const noexcept
- template<typename _InputIterator>
  __detail::_Path2< _InputIterator > & **concat** (_InputIterator __first, _InputIterator __last)
- template<typename _Source>
  __detail::_Path< _Source > & **concat** (_Source const &__x)
- bool **empty** () const noexcept
- iterator **end** () const noexcept
- path **extension** () const

- path **filename** () const
- std::string **generic_string** () const
- template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>>
  std::basic_string< _CharT, _Traits, _Allocator > **generic_string** (const _Allocator &__a=_Allocator()) const
- std::u16string **generic_u16string** () const
- std::u32string **generic_u32string** () const
- std::string **generic_u8string** () const
- std::wstring **generic_wstring** () const
- bool **has_extension** () const noexcept
- bool **has_filename** () const noexcept
- bool **has_parent_path** () const noexcept
- bool **has_relative_path** () const noexcept
- bool **has_root_directory** () const noexcept
- bool **has_root_name** () const noexcept
- bool **has_root_path** () const noexcept
- bool **has_stem** () const noexcept
- bool **is_absolute** () const noexcept
- bool **is_relative** () const noexcept
- path **lexically_normal** () const
- path **lexically_proximate** (const path &base) const
- path **lexically_relative** (const path &base) const
- path & **make_preferred** ()
- const string_type & **native** () const noexcept
- **operator string_type** () const
- template<typename _CharT>
  __detail::_Path2< _CharT * > & **operator+=** (_CharT __x)
- template<typename _Source>
  __detail::_Path< _Source > & **operator+=** (_Source const &__x)
- path & **operator+=** (basic_string_view< value_type > __x)
- path & **operator+=** (const path &__x)
- path & **operator+=** (const string_type &__x)
- path & **operator+=** (const value_type *__x)
- path & **operator+=** (value_type __x)
- template<typename _Source>
  __detail::_Path< _Source > & **operator/=** (_Source const &__source)
- path & **operator/=** (const path &__p)
- template<typename _Source>
  __detail::_Path< _Source > & **operator=** (_Source const &__source)
- path & **operator=** (const path &)
- path & **operator=** (path &&) noexcept
- path & **operator=** (string_type &&__source)
- path **parent_path** () const
- path **relative_path** () const
- path & **remove_filename** ()
- path & **replace_extension** (const path &__replacement=path())
- path & **replace_filename** (const path &__replacement)
- path **root_directory** () const
- path **root_name** () const
- path **root_path** () const
- path **stem** () const
- std::string **string** () const

- template<typename _CharT, typename _Traits, typename _Allocator>
  basic_string< _CharT, _Traits, _Allocator > **string** (const _Allocator &__a) const
- template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>>
  std::basic_string< _CharT, _Traits, _Allocator > **string** (const _Allocator &__a=_Allocator()) const
- void **swap** (path &__rhs) noexcept
- std::u16string **u16string** () const
- std::u32string **u32string** () const
- std::string **u8string** () const
- std::wstring **wstring** () const

Static Public Attributes

- static constexpr value_type **preferred_separator**

Friends

- path operator/ (const path &__lhs, const path &__rhs)
- template<typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const path &__p)
- strong_ordering operator<=> (const path &__lhs, const path &__rhs) noexcept
- bool operator== (const path &__lhs, const path &__rhs) noexcept
- template<typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, path &__p)

Related Symbols

(Note that these are not member symbols.)

- void **swap** (path &__lhs, path &__rhs) noexcept
- template<typename _InputIterator, typename _Require = __detail::_Path2<_InputIterator>, typename _CharT = __detail::__value_type_↩
  is_char_or_char8_t<_InputIterator>>
  path u8path (_InputIterator __first, _InputIterator __last)
- template<typename _Source, typename _Require = __detail::_Path<_Source>, typename _CharT = __detail::__value_type_is_char_or↩
  _char8_t<_Source>>
  path u8path (const _Source &__source)

5.781.1 Detailed Description

A filesystem path.

Since

C++17

5.781.2 Member Enumeration Documentation

format

```
enum std::filesystem::path::format :  unsigned char
```

path::format is ignored in this implementation

5.781.3 Friends And Related Symbol Documentation

operator/

```
path operator/ (
            const path & __lhs,
            const path & __rhs)  [friend]
```

Append one path to another.

operator<<

```
template<typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const path & __p)  [friend]
```

Write a path to a stream.

operator<=>

```
strong_ordering operator<=> (
            const path & __lhs,
            const path & __rhs)  [friend]
```

Compare paths.

operator==

```
bool operator== (
            const path & __lhs,
            const path & __rhs)  [friend]
```

Compare paths.

operator>>

```
template<typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            path & __p)  [friend]
```

Read a path from a stream.
References std::move().
The documentation for this class was generated from the following file:

- bits/fs_path.h

5.782 std::piecewise_constant_distribution< _RealType > Class Template Reference

```
#include <random>
```

Classes

- struct param_type

Public Types

- typedef _RealType result_type

Public Member Functions

- template<typename _InputIteratorB, typename _InputIteratorW>
  piecewise_constant_distribution (_InputIteratorB __bfirst, _InputIteratorB __bend, _InputIteratorW __wbegin)
- **piecewise_constant_distribution** (const param_type &__p)
- template<typename _Func>
  piecewise_constant_distribution (initializer_list< _RealType > __bl, _Func __fw)
- template<typename _Func>
  piecewise_constant_distribution (size_t __nw, _RealType __xmin, _RealType __xmax, _Func __fw)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- std::vector< double > densities () const
- std::vector< _RealType > intervals () const
- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- template<typename _RealType1, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::piecewise_constant_distribution< _RealType1 > &__x)
- bool operator== (const piecewise_constant_distribution &__d1, const piecewise_constant_distribution &__d2)
- template<typename _RealType1, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::piecewise_constant_distribution< _RealType1 > &__x)

5.782.1 Detailed Description

template<typename _RealType = double>
class std::piecewise_constant_distribution< _RealType >

A piecewise_constant_distribution random number distribution.
This distribution produces random numbers $x, b_0 \leq x < b_n$, uniformly distributed over each subinterval $[b_i, b_{i+1})$ according to the probability mass function

$$p(x|b_0, ..., b_n, \rho_0, ..., \rho_{n-1}) = \rho_i \cdot \frac{b_{i+1} - x}{b_{i+1} - b_i} + \rho_{i+1} \cdot \frac{x - b_i}{b_{i+1} - b_i}$$

for $b_i \leq x < b_{i+1}$.

Since

C++11

5.782.2 Member Typedef Documentation

result_type

```
template<typename _RealType = double>
typedef _RealType std::piecewise_constant_distribution< _RealType >::result_type
```

The type of the range of the distribution.

5.782.3 Member Function Documentation

densities()

```
template<typename _RealType = double>
std::vector< double > std::piecewise_constant_distribution< _RealType >::densities () const
[inline]
```

Returns a vector of the probability densities.

intervals()

```
template<typename _RealType = double>
std::vector< _RealType > std::piecewise_constant_distribution< _RealType >::intervals () const
[inline]
```

Returns a vector of the intervals.

max()

```
template<typename _RealType = double>
result_type std::piecewise_constant_distribution< _RealType >::max () const  [inline]
```

Returns the least upper bound value of the distribution.

min()

```
template<typename _RealType = double>
result_type std::piecewise_constant_distribution< _RealType >::min () const  [inline]
```

Returns the greatest lower bound value of the distribution.

operator()()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::piecewise_constant_distribution< _RealType >::operator() (
            _UniformRandomNumberGenerator & __urng)  [inline]
```

Generating functions.
Referenced by operator()().

param() [1/2]

```
template<typename _RealType = double>
param_type std::piecewise_constant_distribution< _RealType >::param () const  [inline]
```

Returns the parameter set of the distribution.

param() [2/2]

```
template<typename _RealType = double>
void std::piecewise_constant_distribution< _RealType >::param (
            const param_type & __param)  [inline]
```

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

reset()

```
template<typename _RealType = double>
void std::piecewise_constant_distribution< _RealType >::reset ()  [inline]
```

Resets the distribution state.

5.782.4 Friends And Related Symbol Documentation

operator<<

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::piecewise_constant_distribution< _RealType1 > & __x)  [friend]
```

Inserts a piecewise_constant_distribution random number distribution __x into the output stream __os.

Parameters

__os	An output stream.
__x	A piecewise_constant_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

operator==

```
template<typename _RealType = double>
bool operator== (
            const piecewise_constant_distribution< _RealType > & __d1,
            const piecewise_constant_distribution< _RealType > & __d2)  [friend]
```

Return true if two piecewise constant distributions have the same parameters.

operator>>

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            std::piecewise_constant_distribution< _RealType1 > & __x)  [friend]
```

Extracts a piecewise_constant_distribution random number distribution __x from the input stream __is.

Parameters

_↩ _is	An input stream.
_↩ _x	A piecewise_constant_distribution random number generator engine.

Returns

The input stream with __x extracted or in an error state.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

5.783 std::piecewise_construct_t Struct Reference

`#include <stl_pair.h>`

5.783.1 Detailed Description

Tag type for piecewise construction of std::pair objects.
The documentation for this struct was generated from the following file:

- stl_pair.h

5.784 std::piecewise_linear_distribution< _RealType > Class Template Reference

`#include <random>`

Classes

- struct param_type

Public Types

- typedef _RealType result_type

Public Member Functions

- template<typename _InputIteratorB, typename _InputIteratorW>
  piecewise_linear_distribution (_InputIteratorB __bfirst, _InputIteratorB __bend, _InputIteratorW __wbegin)
- **piecewise_linear_distribution** (const param_type &__p)
- template<typename _Func>
  piecewise_linear_distribution (initializer_list< _RealType > __bl, _Func __fw)
- template<typename _Func>
  piecewise_linear_distribution (size_t __nw, _RealType __xmin, _RealType __xmax, _Func __fw)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- std::vector< double > densities () const
- std::vector< _RealType > intervals () const
- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng)

- template<typename _UniformRandomNumberGenerator>
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- template<typename _RealType1, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::piecewise_linear_distribution< _RealType1 > &__x)
- bool operator== (const piecewise_linear_distribution &__d1, const piecewise_linear_distribution &__d2)
- template<typename _RealType1, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::piecewise_linear_distribution< _RealType1 > &__x)

5.784.1 Detailed Description

template<typename _RealType = double>
class std::piecewise_linear_distribution< _RealType >

A piecewise_linear_distribution random number distribution.
This distribution produces random numbers $x, b_0 \leq x < b_n$, distributed over each subinterval $[b_i, b_{i+1})$ according to the probability mass function $p(x|b_0, ..., b_n, \rho_0, ..., \rho_n) = \rho_i$, for $b_i \leq x < b_{i+1}$.

Since

C++11

5.784.2 Member Typedef Documentation

result_type

```
template<typename _RealType = double>
typedef _RealType std::piecewise_linear_distribution< _RealType >::result_type
```

The type of the range of the distribution.

5.784.3 Member Function Documentation

densities()

```
template<typename _RealType = double>
std::vector< double > std::piecewise_linear_distribution< _RealType >::densities () const  [inline]
```

Return a vector of the probability densities of the distribution.

intervals()

```
template<typename _RealType = double>
std::vector< _RealType > std::piecewise_linear_distribution< _RealType >::intervals () const
[inline]
```

Return the intervals of the distribution.

max()

```
template<typename _RealType = double>
result_type std::piecewise_linear_distribution< _RealType >::max () const  [inline]
```

Returns the least upper bound value of the distribution.

min()

```
template<typename _RealType = double>
result_type std::piecewise_linear_distribution< _RealType >::min () const  [inline]
```

Returns the greatest lower bound value of the distribution.

operator()()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::piecewise_linear_distribution< _RealType >::operator() (
            _UniformRandomNumberGenerator & __urng)  [inline]
```

Generating functions.
Referenced by operator()().

param() [1/2]

```
template<typename _RealType = double>
param_type std::piecewise_linear_distribution< _RealType >::param () const  [inline]
```

Returns the parameter set of the distribution.

param() [2/2]

```
template<typename _RealType = double>
void std::piecewise_linear_distribution< _RealType >::param (
            const param_type & __param)  [inline]
```

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

reset()

```
template<typename _RealType = double>
void std::piecewise_linear_distribution< _RealType >::reset ()  [inline]
```

Resets the distribution state.

5.784.4 Friends And Related Symbol Documentation

operator<<

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::piecewise_linear_distribution< _RealType1 > & __x)  [friend]
```

Inserts a piecewise_linear_distribution random number distribution __x into the output stream __os.

Parameters

__os	An output stream.
__x	A piecewise_linear_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

operator==

```
template<typename _RealType = double>
bool operator== (
             const piecewise_linear_distribution< _RealType > & __d1,
             const piecewise_linear_distribution< _RealType > & __d2)  [friend]
```

Return true if two piecewise linear distributions have the same parameters.

operator>>

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
             std::basic_istream< _CharT, _Traits > & __is,
             std::piecewise_linear_distribution< _RealType1 > & __x)  [friend]
```

Extracts a piecewise_linear_distribution random number distribution __x from the input stream __is.

Parameters

_↩ _is	An input stream.
_↩ _x	A piecewise_linear_distribution random number generator engine.

Returns

The input stream with __x extracted or in an error state.

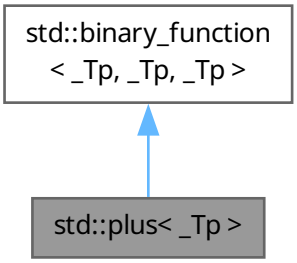
The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

5.785 std::plus< _Tp > Struct Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::plus< _Tp >:



Public Types

- typedef _Tp first_argument_type
- typedef _Tp result_type
- typedef _Tp second_argument_type

Public Member Functions

- constexpr _Tp operator() (const _Tp &__x, const _Tp &__y) const

5.785.1 Detailed Description

template<typename _Tp>
struct std::plus< _Tp >

One of the math functors.

5.785.2 Member Typedef Documentation

first_argument_type

```
typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::first_argument_type  [inherited]
```

`first_argument_type` is the type of the first argument

result_type

```
typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::result_type  [inherited]
```

`result_type` is the return type

second_argument_type

```
typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::second_argument_type  [inherited]
```

`second_argument_type` is the type of the second argument

5.785.3 Member Function Documentation

operator()()

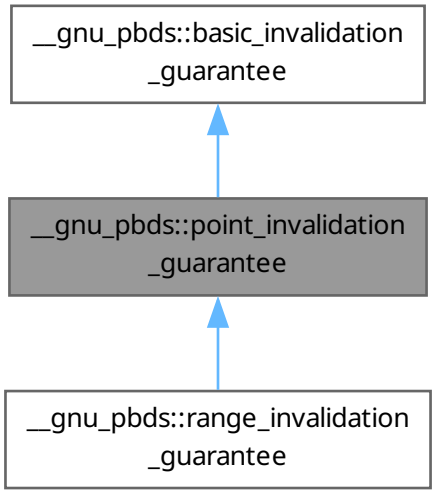
```
template<typename _Tp>
_Tp std::plus< _Tp >::operator() (
            const _Tp & __x,
            const _Tp & __y) const  [inline], [constexpr]
```

Returns the sum.
The documentation for this struct was generated from the following file:

- stl_function.h

5.786 __gnu_pbds::point_invalidation_guarantee Struct Reference

`#include <tag_and_trait.hpp>`
Inheritance diagram for __gnu_pbds::point_invalidation_guarantee:



5.786.1 Detailed Description

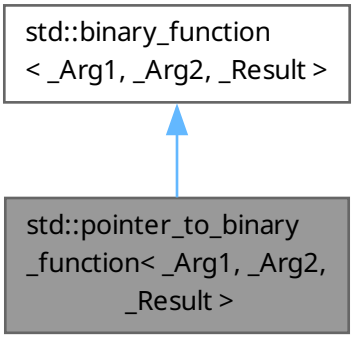
Signifies an invalidation guarantee that includes all those of its base, and additionally, that any point-type iterator, pointer, or reference to a container object's mapped value type is valid as long as its corresponding entry has not be erased, regardless of modifications to the container object.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

5.787 std::pointer_to_binary_function< _Arg1, _Arg2, _Result > Class Template Reference

`#include <stl_function.h>`

Inheritance diagram for std::pointer_to_binary_function< _Arg1, _Arg2, _Result >:



Public Types

- typedef _Arg1 first_argument_type
- typedef _Result result_type
- typedef _Arg2 second_argument_type

Public Member Functions

- **pointer_to_binary_function** (_Result(∗__x)(_Arg1, _Arg2))
- _Result **operator()** (_Arg1 __x, _Arg2 __y) const

Protected Attributes

- _Result(∗ **_M_ptr**)(_Arg1, _Arg2)

5.787.1 Detailed Description

template<typename _Arg1, typename _Arg2, typename _Result>
class std::pointer_to_binary_function< _Arg1, _Arg2, _Result >

One of the adaptors for function pointers.

5.787.2 Member Typedef Documentation

first_argument_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result >::first_argument_type  [inherited]
```

`first_argument_type` is the type of the first argument

result_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Result std::binary_function< _Arg1, _Arg2, _Result >::result_type  [inherited]
```

`result_type` is the return type

second_argument_type

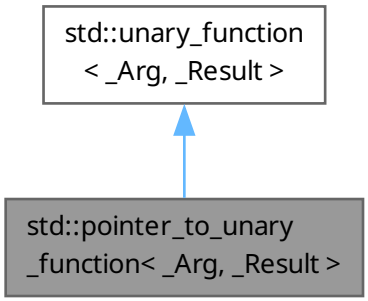
```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result >::second_argument_type  [inherited]
```

second_argument_type is the type of the second argument
The documentation for this class was generated from the following file:

- stl_function.h

5.788 std::pointer_to_unary_function< _Arg, _Result > Class Template Reference

`#include <stl_function.h>`
Inheritance diagram for std::pointer_to_unary_function< _Arg, _Result >:



Public Types

- typedef _Arg argument_type
- typedef _Result result_type

Public Member Functions

- **pointer_to_unary_function** (_Result(*__x)(_Arg))
- _Result **operator()** (_Arg __x) const

Protected Attributes

- _Result(* **_M_ptr**)(_Arg)

5.788.1 Detailed Description

template<typename _Arg, typename _Result>
class std::pointer_to_unary_function< _Arg, _Result >

One of the adaptors for function pointers.

5.788.2 Member Typedef Documentation

argument_type

```
template<typename _Arg, typename _Result>
typedef _Arg std::unary_function< _Arg, _Result >::argument_type  [inherited]
```

argument_type is the type of the argument

result_type

```
template<typename _Arg, typename _Result>
typedef _Result std::unary_function< _Arg, _Result >::result_type  [inherited]
```

result_type is the return type

The documentation for this class was generated from the following file:

- stl_function.h

5.789 std::pointer_traits< _Ptr > Struct Template Reference

```
#include <memory>
```

Public Types

- using difference_type
- using element_type
- using pointer
- template<typename _Up>
  using rebind

Static Public Member Functions

- static pointer pointer_to (element_type &__r)

5.789.1 Detailed Description

template<typename _Ptr>
struct std::pointer_traits< _Ptr >

Uniform interface to all pointer-like types.

Since

C++11

5.789.2 Member Typedef Documentation

difference_type

```
template<typename _Ptr, typename _Elt>
using std::__ptr_traits_impl< _Ptr, _Elt >::difference_type  [inherited]
```

The type used to represent the difference between two pointers.

element_type

```
template<typename _Ptr, typename _Elt>
using std::__ptr_traits_impl< _Ptr, _Elt >::element_type  [inherited]
```

The type pointed to.

pointer

```
template<typename _Ptr, typename _Elt>
using std::__ptr_traits_impl< _Ptr, _Elt >::pointer  [inherited]
```
The pointer type.

rebind

```
template<typename _Ptr, typename _Elt>
template<typename _Up>
using std::__ptr_traits_impl< _Ptr, _Elt >::rebind  [inherited]
```
A pointer to a different type.

5.789.3 Member Function Documentation

pointer_to()

```
template<typename _Ptr, typename _Elt, bool = is_void<_Elt>::value>
static pointer std::__ptr_traits_ptr_to< _Ptr, _Elt, bool >::pointer_to (
            element_type & __r)  [inline], [static], [inherited]
```
Obtain a pointer to an object.

Parameters

↩ _↩ ↩ _↩ *r*	A reference to an object of type `element_type`

Returns

`pointer::pointer_to(__r)`

Precondition

`pointer::pointer_to(__r)` is a valid expression.

The documentation for this struct was generated from the following file:

- ptr_traits.h

5.790 std::pointer_traits< _Tp * > Struct Template Reference

`#include <memory>`

Public Types

- using difference_type
- typedef ptrdiff_t difference_type
- using element_type
- typedef _Tp element_type
- using pointer
- typedef _Tp * pointer
- using rebind
- template<typename _Up>
  using rebind

Static Public Member Functions

- static pointer pointer_to (element_type &__r)
- static pointer pointer_to (element_type &__r)
- static pointer pointer_to (element_type &__r)

5.790.1 Detailed Description

template<typename _Tp>
struct std::pointer_traits< _Tp * >

Partial specialization for built-in pointers.

Since

C++11

5.790.2 Member Typedef Documentation

difference_type [1/2]

```
using std::__ptr_traits_impl< _Tp *, _Elt >::difference_type
```
The type used to represent the difference between two pointers.

difference_type [2/2]

```
template<typename _Tp>
typedef ptrdiff_t std::pointer_traits< _Tp * >::difference_type
```
Type used to represent the difference between two pointers.

element_type [1/2]

```
using std::__ptr_traits_impl< _Tp *, _Elt >::element_type
```
The type pointed to.

element_type [2/2]

```
template<typename _Tp>
typedef _Tp std::pointer_traits< _Tp * >::element_type
```
The type pointed to.

pointer [1/2]

```
using std::__ptr_traits_impl< _Tp *, _Elt >::pointer
```
The pointer type.

pointer [2/2]

```
template<typename _Tp>
typedef _Tp* std::pointer_traits< _Tp * >::pointer
```
The pointer type.

rebind [1/2]

```
using std::__ptr_traits_impl< _Tp *, _Elt >::rebind
```
A pointer to a different type.

rebind [2/2]

```
template<typename _Tp>
template<typename _Up>
using std::pointer_traits< _Tp * >::rebind
```

A pointer to a different type.

5.790.3 Member Function Documentation

pointer_to() [1/3]

```
template<typename _Ptr, typename _Elt, bool = is_void<_Elt>::value>
static pointer std::__ptr_traits_ptr_to< _Ptr, _Elt, bool >::pointer_to (
            element_type & __r)  [inline], [static], [inherited]
```

Obtain a pointer to an object.

Parameters

↩ _↩ ↩ _↩ *r*	A reference to an object of type `element_type`

Returns

`pointer::pointer_to(__r)`

Precondition

`pointer::pointer_to(__r)` is a valid expression.

pointer_to() [2/3]

```
template<typename _Ptr, typename _Elt, bool = is_void<_Elt>::value>
static pointer std::__ptr_traits_ptr_to< _Ptr, _Elt, bool >::pointer_to (
            element_type & __r)  [inline], [static], [inherited]
```

Obtain a pointer to an object.

Parameters

↩ _↩ ↩ _↩ *r*	A reference to an object of type `element_type`

Returns

`pointer::pointer_to(__r)`

Precondition

`pointer::pointer_to(__r)` is a valid expression.

pointer_to() [3/3]

```
static pointer std::__ptr_traits_ptr_to< _Tp *, _Elt, bool >::pointer_to (
            element_type & __r)  [inline], [static]
```

Obtain a pointer to an object.

Parameters

↩ _↩ ↩ _↩ *r*	A reference to an object of type `element_type`

Returns

`pointer::pointer_to(__r)`

Precondition

`pointer::pointer_to(__r)` is a valid expression.

The documentation for this struct was generated from the following file:

- ptr_traits.h

5.791 std::poisson_distribution< _IntType > Class Template Reference

`#include <random>`

Classes

- struct param_type

Public Types

- typedef _IntType result_type

Public Member Functions

- **poisson_distribution** (const param_type &__p)
- **poisson_distribution** (double __mean)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- result_type max () const
- double mean () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng)

- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- template<typename _IntType1, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::poisson_distribution< _IntType1 > &__x)
- bool operator== (const poisson_distribution &__d1, const poisson_distribution &__d2)
- template<typename _IntType1, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::poisson_distribution< _IntType1 > &__x)

5.791.1 Detailed Description

template<typename _IntType = int>
class std::poisson_distribution< _IntType >

A discrete Poisson random number distribution.
The formula for the Poisson probability density function is $p(i|\mu) = \frac{\mu^i}{i!}e^{-\mu}$ where μ is the parameter of the distribution.

Since

C++11

5.791.2 Member Typedef Documentation

result_type

```
template<typename _IntType = int>
typedef _IntType std::poisson_distribution< _IntType >::result_type
```

The type of the range of the distribution.

5.791.3 Member Function Documentation

max()

```
template<typename _IntType = int>
result_type std::poisson_distribution< _IntType >::max () const  [inline]
```

Returns the least upper bound value of the distribution.

mean()

```
template<typename _IntType = int>
double std::poisson_distribution< _IntType >::mean () const  [inline]
```

Returns the distribution parameter `mean`.

min()

```
template<typename _IntType = int>
result_type std::poisson_distribution< _IntType >::min () const  [inline]
```

Returns the greatest lower bound value of the distribution.

operator()() [1/2]

```
template<typename _IntType = int>
template<typename _UniformRandomNumberGenerator>
result_type std::poisson_distribution< _IntType >::operator() (
            _UniformRandomNumberGenerator & __urng)  [inline]
```

Generating functions.
Referenced by operator()().

operator()() [2/2]

```
template<typename _IntType>
template<typename _UniformRandomNumberGenerator>
poisson_distribution< _IntType >::result_type std::poisson_distribution< _IntType >::operator()
(
            _UniformRandomNumberGenerator & __urng,
            const param_type & __param)
```

A rejection algorithm when mean >= 12 and a simple method based upon the multiplication of uniform random variates otherwise. NB: The former is available only if _GLIBCXX_USE_C99_MATH_FUNCS is defined.
Reference: Devroye, L. Non-Uniform Random Variates Generation. Springer-Verlag, New York, 1986, Ch. X, Sects. 3.3 & 3.4 (+ Errata!).
References std::abs(), std::numeric_limits< _Tp >::epsilon(), std::log(), and std::numeric_limits< _Tp >::max().

param() [1/2]

```
template<typename _IntType = int>
param_type std::poisson_distribution< _IntType >::param () const  [inline]
```

Returns the parameter set of the distribution.

param() [2/2]

```
template<typename _IntType = int>
void std::poisson_distribution< _IntType >::param (
            const param_type & __param)  [inline]
```

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

reset()

```
template<typename _IntType = int>
void std::poisson_distribution< _IntType >::reset ()  [inline]
```

Resets the distribution state.

5.791.4 Friends And Related Symbol Documentation

operator<<

```
template<typename _IntType = int>
template<typename _IntType1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::poisson_distribution< _IntType1 > & __x)  [friend]
```

Inserts a poisson_distribution random number distribution __x into the output stream __os.

Parameters

__os	An output stream.
__x	A poisson_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

operator==

```
template<typename _IntType = int>
bool operator== (
             const poisson_distribution< _IntType > & __d1,
             const poisson_distribution< _IntType > & __d2)  [friend]
```

Return true if two Poisson distributions have the same parameters and the sequences that would be generated are equal.

operator>>

```
template<typename _IntType = int>
template<typename _IntType1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
             std::basic_istream< _CharT, _Traits > & __is,
             std::poisson_distribution< _IntType1 > & __x)  [friend]
```

Extracts a poisson_distribution random number distribution __x from the input stream __is.

Parameters

_↩ _is	An input stream.
_↩ _x	A poisson_distribution random number generator engine.

Returns

The input stream with __x extracted or in an error state.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

5.792 std::pmr::polymorphic_allocator< _Tp > Class Template Reference

```
#include <memory_resource>
```

Public Types

- using **value_type**

Public Member Functions

- **polymorphic_allocator** (const polymorphic_allocator &__other)=default
- template<typename _Up>
  polymorphic_allocator (const polymorphic_allocator< _Up > &__x) noexcept
- **polymorphic_allocator** (memory_resource *__r) noexcept
- _Tp * **allocate** (size_t __n)
- template<typename _Tp1, typename... _Args>
  void **construct** (_Tp1 *__p, _Args &&... __args)
- void **deallocate** (_Tp *__p, size_t __n) noexcept
- template<typename _Up>
  void **destroy** (_Up *__p)
- polymorphic_allocator & **operator=** (const polymorphic_allocator &)=delete
- memory_resource * **resource** () const noexcept
- polymorphic_allocator **select_on_container_copy_construction** () const noexcept

Friends

- bool **operator==** (const polymorphic_allocator &__a, const polymorphic_allocator &__b) noexcept

5.792.1 Detailed Description

template<typename _Tp>
class std::pmr::polymorphic_allocator< _Tp >

Class template polymorphic_allocator.

Since

C++17

The documentation for this class was generated from the following file:

- memory_resource.h

5.793 std::pmr::pool_options Struct Reference

`#include <memory_resource>`

Public Attributes

- size_t **largest_required_pool_block**
- size_t max_blocks_per_chunk

5.793.1 Detailed Description

Parameters for tuning a pool resource's behaviour.

Since

C++17

5.793.2 Member Data Documentation

max_blocks_per_chunk

`size_t std::pmr::pool_options::max_blocks_per_chunk`
Upper limit on number of blocks in a chunk.
A lower value prevents allocating huge chunks that could remain mostly unused, but means pools will need to replenished more frequently.
The documentation for this struct was generated from the following file:

- memory_resource

5.794 __gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc > Class Template Reference

`#include <priority_queue.hpp>`

Public Types

- typedef _Alloc **allocator_type**
- typedef Cmp_Fn **cmp_fn**
- typedef base_type::const_iterator **const_iterator**
- typedef __rebind_va::const_pointer **const_pointer**
- typedef __rebind_va::const_reference **const_reference**
- typedef Tag **container_category**
- typedef allocator_type::difference_type **difference_type**
- typedef base_type::iterator **iterator**
- typedef base_type::point_const_iterator **point_const_iterator**
- typedef base_type::point_iterator **point_iterator**
- typedef __rebind_va::pointer **pointer**
- typedef __rebind_va::reference **reference**
- typedef allocator_type::size_type **size_type**
- typedef _Tv **value_type**

Public Member Functions

- priority_queue (const cmp_fn &r_cmp_fn)
- **priority_queue** (const priority_queue &other)
- template<typename It>
  priority_queue (It first_it, It last_it)
- template<typename It>
  priority_queue (It first_it, It last_it, const cmp_fn &r_cmp_fn)
- priority_queue & **operator=** (const priority_queue &other)
- void **swap** (priority_queue &other)

5.794.1 Detailed Description

template<typename _Tv, typename Cmp_Fn = std::less<_Tv>, typename Tag = pairing_heap_tag, typename _Alloc = std::allocator<char>>
class __gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >

A priority queue composed of one specific heap policy.

Template Parameters

_Tv	Value type.
Cmp_Fn	Comparison functor.
Tag	Instantiating data structure type, see container_tag.
_Alloc	Allocator type.

Base is dispatched at compile time via Tag, from the following choices: binary_heap_tag, binomial_heap_tag, pairing_heap_tag, rc_binomial_heap_tag, thin_heap_tag
Base choices are: detail::binary_heap, detail::binomial_heap, detail::pairing_heap, detail::rc_binomial_heap, detail::thin_heap.

5.794.2 Constructor & Destructor Documentation

priority_queue() [1/3]

```
template<typename _Tv, typename Cmp_Fn = std::less<_Tv>, typename Tag = pairing_heap_tag, typename
_Alloc = std::allocator<char>>
__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::priority_queue (
            const cmp_fn & r_cmp_fn)  [inline]
```

Constructor taking some policy objects. r_cmp_fn will be copied by the Cmp_Fn object of the container object.

priority_queue() [2/3]

```
template<typename _Tv, typename Cmp_Fn = std::less<_Tv>, typename Tag = pairing_heap_tag, typename
_Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::priority_queue (
            It first_it,
            It last_it)  [inline]
```

Constructor taking __iterators to a range of value_types. The value_types between first_it and last_it will be inserted into the container object.

priority_queue() [3/3]

```
template<typename _Tv, typename Cmp_Fn = std::less<_Tv>, typename Tag = pairing_heap_tag, typename
_Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::priority_queue (
            It first_it,
            It last_it,
            const cmp_fn & r_cmp_fn)  [inline]
```

Constructor taking __iterators to a range of value_types and some policy objects The value_types between first_it and last_it will be inserted into the container object. r_cmp_fn will be copied by the cmp_fn object of the container object.
The documentation for this class was generated from the following file:

- priority_queue.hpp

5.795 std::priority_queue< _Tp, _Sequence, _Compare > Class Template Reference

```
#include <stl_queue.h>
```

Public Types

- typedef _Sequence::const_reference **const_reference**

- typedef _Sequence **container_type**
- typedef _Sequence::reference **reference**
- typedef _Sequence::size_type **size_type**
- typedef _Compare **value_compare**
- typedef _Sequence::value_type **value_type**

Public Member Functions

- template<typename _Seq = _Sequence, typename _Requires = typename enable_if<__and_<is_default_constructible<_Compare>, is↩ _default_constructible<_Seq>>::value>::type>
  priority_queue ()
- template<typename _InputIterator, typename _Alloc, typename = std::_RequireInputIter<_InputIterator>, typename _Requires = _Uses<↩ _Alloc>>
  priority_queue (_InputIterator __first, _InputIterator __last, const _Alloc &__alloc)
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  priority_queue (_InputIterator __first, _InputIterator __last, const _Compare &__x, _Sequence &&__s)
- template<typename _InputIterator, typename _Alloc, typename _Requires = _Uses<_Alloc>>
  priority_queue (_InputIterator __first, _InputIterator __last, const _Compare &__x, _Sequence &&__s, const _Alloc &__alloc)
- template<typename _InputIterator, typename _Alloc, typename = std::_RequireInputIter<_InputIterator>, typename _Requires = _Uses<↩ _Alloc>>
  priority_queue (_InputIterator __first, _InputIterator __last, const _Compare &__x, const _Alloc &__alloc)
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  priority_queue (_InputIterator __first, _InputIterator __last, const _Compare &__x, const _Sequence &__s)
- template<typename _InputIterator, typename _Alloc, typename = std::_RequireInputIter<_InputIterator>, typename _Requires = _Uses<↩ _Alloc>>
  priority_queue (_InputIterator __first, _InputIterator __last, const _Compare &__x, const _Sequence &__s, const _Alloc &__alloc)
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  priority_queue (_InputIterator __first, _InputIterator __last, const _Compare &__x=_Compare())
- template<typename _Alloc, typename _Requires = _Uses<_Alloc>>
  priority_queue (const _Alloc &__a)
- template<typename _Alloc, typename _Requires = _Uses<_Alloc>>
  priority_queue (const _Compare &__x, _Sequence &&__c, const _Alloc &__a)
- **priority_queue** (const _Compare &__x, _Sequence &&__s=_Sequence())
- template<typename _Alloc, typename _Requires = _Uses<_Alloc>>
  priority_queue (const _Compare &__x, const _Alloc &__a)
- template<typename _Alloc, typename _Requires = _Uses<_Alloc>>
  priority_queue (const _Compare &__x, const _Sequence &__c, const _Alloc &__a)
- **priority_queue** (const _Compare &__x, const _Sequence &__s)
- **priority_queue** (const priority_queue &)=default
- template<typename _Alloc, typename _Requires = _Uses<_Alloc>>
  priority_queue (const priority_queue &__q, const _Alloc &__a)
- **priority_queue** (priority_queue &&__q) noexcept(__and_< is_nothrow_move_constructible< _Sequence >, is_nothrow_move_constructible< _Compare > >::value)
- template<typename _Alloc, typename _Requires = _Uses<_Alloc>>
  priority_queue (priority_queue &&__q, const _Alloc &__a)
- template<typename... _Args>
  void **emplace** (_Args &&... __args)
- bool empty () const
- priority_queue & **operator=** (const priority_queue &)=default
- priority_queue & **operator=** (priority_queue &&__q) noexcept(__and_< is_nothrow_move_assignable< _↩ Sequence >, is_nothrow_move_assignable< _Compare > >::value)

- void pop ()
- void push (const value_type &__x)
- void **push** (value_type &&__x)
- size_type size () const
- void **swap** (priority_queue &__pq) noexcept(__and_<//c++1z or gnu++11 __is_nothrow_swappable< _Sequence >, __is_nothrow_swappable< _Compare > >::value)
- const_reference top () const

Protected Attributes

- _Sequence **c**
- _Compare **comp**

5.795.1 Detailed Description

template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _Sequence::value_type>>
class std::priority_queue< _Tp, _Sequence, _Compare >

A standard container automatically sorting its contents.

Template Parameters

_Tp	Type of element.
_Sequence	Type of underlying sequence, defaults to vector<_Tp>.
_Compare	Comparison function object type, defaults to less<_Sequence::value_type>.

This is not a true container, but an *adaptor*. It holds another container, and provides a wrapper interface to that container. The wrapper is what enforces priority-based sorting and queue behavior. Very few of the standard container/sequence interface requirements are met (e.g., iterators).
The second template parameter defines the type of the underlying sequence/container. It defaults to std::vector, but it can be any type that supports `front()`, `push_back`, `pop_back`, and random-access iterators, such as std::deque or an appropriate user-defined type.
The third template parameter supplies the means of making priority comparisons. It defaults to `less<value_type>` but can be anything defining a strict weak ordering.
Members not found in *normal* containers are `container_type`, which is a typedef for the second Sequence parameter, and `push`, `pop`, and `top`, which are standard queue operations.

Note

No equality/comparison operators are provided for priority_queue.

Sorting of the elements takes place as they are added to, and removed from, the priority_queue using the priority_queue's member functions. If you access the elements by other means, and change their data such that the sorting order would be different, the priority_queue will not re-sort the elements for you. (How could it know to do so?)

5.795.2 Constructor & Destructor Documentation

priority_queue() [1/2]

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _Sequence::value_type>>
template<typename _Seq = _Sequence, typename _Requires = typename enable_if<__and_<is_default_constructible<_Compare>, is_default_constructible<_Seq>>::value>::type>
std::priority_queue< _Tp, _Sequence, _Compare >::priority_queue ()  [inline]
```

Default constructor creates no elements.

priority_queue() [2/2]

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _↩
Sequence::value_type>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
std::priority_queue< _Tp, _Sequence, _Compare >::priority_queue (
            _InputIterator __first,
            _InputIterator __last,
            const _Compare & __x = _Compare())  [inline]
```

Builds a queue from a range.

Parameters

__first	An input iterator.
__last	An input iterator.
__x	A comparison functor describing a strict weak ordering.
__s	An initial sequence with which to start.

Begins by copying *__s*, inserting a copy of the elements from [first,last) into the copy of *__s*, then ordering the copy according to *__x*.
For more information on function objects, see the documentation on functor base classes.

5.795.3 Member Function Documentation

empty()

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _↩
Sequence::value_type>>
bool std::priority_queue< _Tp, _Sequence, _Compare >::empty () const  [inline], [nodiscard]
```

Returns true if the queue is empty.
Referenced by __gnu_parallel::multiseq_partition(), and __gnu_parallel::multiseq_selection().

pop()

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _↩
Sequence::value_type>>
void std::priority_queue< _Tp, _Sequence, _Compare >::pop ()  [inline]
```

Removes first element.
This is a typical queue operation. It shrinks the queue by one. The time complexity of the operation depends on the underlying sequence.
Note that no data is returned, and if the first element's data is needed, it should be retrieved before pop() is called.
Referenced by __gnu_parallel::multiseq_partition(), and __gnu_parallel::multiseq_selection().

push()

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _↩
Sequence::value_type>>
void std::priority_queue< _Tp, _Sequence, _Compare >::push (
            const value_type & __x)  [inline]
```

Add data to the queue.

Parameters

_↩ _x	Data to be added.

This is a typical queue operation. The time complexity of the operation depends on the underlying sequence.
Referenced by __gnu_parallel::multiseq_partition(), and __gnu_parallel::multiseq_selection().

size()

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _↩
Sequence::value_type>>
size_type std::priority_queue< _Tp, _Sequence, _Compare >::size () const  [inline], [nodiscard]
```

Returns the number of elements in the queue.

top()

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _↩
Sequence::value_type>>
const_reference std::priority_queue< _Tp, _Sequence, _Compare >::top () const  [inline], [nodiscard]
```

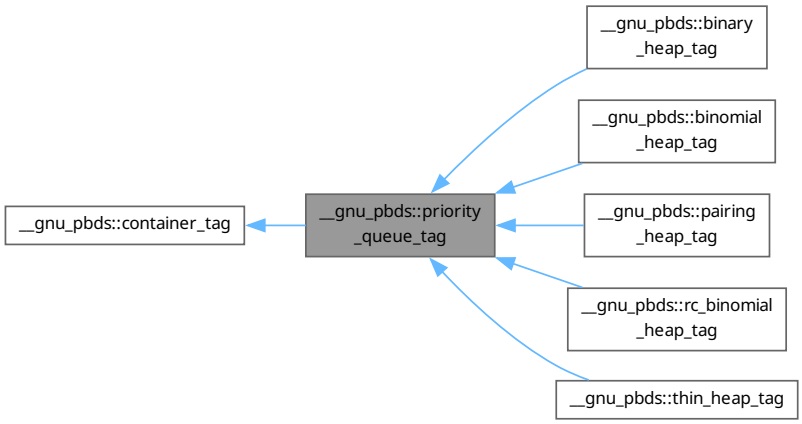
Returns a read-only (constant) reference to the data at the first element of the queue.
Referenced by __gnu_parallel::multiseq_partition(), and __gnu_parallel::multiseq_selection().
The documentation for this class was generated from the following file:

- stl_queue.h

5.796 __gnu_pbds::priority_queue_tag Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for __gnu_pbds::priority_queue_tag:



5.796.1 Detailed Description

Basic priority-queue.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

5.797 __gnu_pbds::detail::probe_fn_base< _Alloc > Class Template Reference

`#include <probe_fn_base.hpp>`

5.797.1 Detailed Description

template<typename _Alloc>
class __gnu_pbds::detail::probe_fn_base< _Alloc >

Probe functor base.
The documentation for this class was generated from the following file:

- probe_fn_base.hpp

5.798 __gnu_cxx::project1st< _Arg1, _Arg2 > Struct Template Reference

`#include <functional>`

Public Types

- typedef _Arg1 first_argument_type
- typedef _Arg1 result_type
- typedef _Arg2 second_argument_type

Public Member Functions

- _Arg1 **operator()** (const _Arg1 &__x, const _Arg2 &) const

5.798.1 Detailed Description

template<class _Arg1, class _Arg2>
struct __gnu_cxx::project1st< _Arg1, _Arg2 >

An SGI extension .

5.798.2 Member Typedef Documentation

first_argument_type

`typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Arg1 >::first_argument_type [inherited]`
`first_argument_type` is the type of the first argument

result_type

`typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Arg1 >::result_type [inherited]`
`result_type` is the return type

second_argument_type

`typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Arg1 >::second_argument_type [inherited]`
`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- ext/functional

5.799 __gnu_cxx::project2nd< _Arg1, _Arg2 > Struct Template Reference

`#include <functional>`

Public Types

- typedef _Arg1 first_argument_type
- typedef _Arg2 result_type
- typedef _Arg2 second_argument_type

Public Member Functions

- _Arg2 **operator()** (const _Arg1 &, const _Arg2 &__y) const

5.799.1 Detailed Description

template<class _Arg1, class _Arg2>
struct __gnu_cxx::project2nd< _Arg1, _Arg2 >

An SGI extension .

5.799.2 Member Typedef Documentation

first_argument_type

`typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Arg2 >::first_argument_type [inherited]`
`first_argument_type` is the type of the first argument

result_type

`typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Arg2 >::result_type [inherited]`
`result_type` is the return type

second_argument_type

`typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Arg2 >::second_argument_type [inherited]`
`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- ext/functional

5.800 std::promise< _Res > Class Template Reference

`#include <future>`

Public Member Functions

- template<typename _Allocator>
  promise (allocator_arg_t, const _Allocator &, promise &&__rhs)
- template<typename _Allocator>
  promise (allocator_arg_t, const _Allocator &__a)
- **promise** (const promise &)=delete
- **promise** (promise &&__rhs) noexcept
- future< _Res > **get_future** ()
- promise & **operator=** (const promise &)=delete
- promise & **operator=** (promise &&__rhs) noexcept
- void **set_exception** (exception_ptr __p)
- void **set_exception_at_thread_exit** (exception_ptr __p)
- void **set_value** (_Res &&__r)
- void **set_value** (const _Res &__r)

- void **set_value_at_thread_exit** (_Res &&__r)
- void **set_value_at_thread_exit** (const _Res &__r)
- void **swap** (promise &__rhs) noexcept

Friends

- template<typename, typename>
  struct **_State::_Setter**

5.800.1 Detailed Description

template<typename _Res>
class std::promise< _Res >

Primary template for promise.
The documentation for this class was generated from the following file:

- future

5.801 std::promise< _Res & > Class Template Reference

`#include <future>`

Public Member Functions

- **promise** (allocator_arg_t, const _Allocator &, promise &&__rhs)
- template<typename _Allocator>
  promise (allocator_arg_t, const _Allocator &, promise &&__rhs)
- **promise** (allocator_arg_t, const _Allocator &__a)
- template<typename _Allocator>
  promise (allocator_arg_t, const _Allocator &__a)
- **promise** (const promise &)=delete
- **promise** (const promise &)=delete
- **promise** (promise &&__rhs) noexcept
- **promise** (promise &&__rhs) noexcept
- future< _Res > **get_future** ()
- future< _Res & > **get_future** ()
- promise & **operator=** (const promise &)=delete
- promise & **operator=** (const promise &)=delete
- promise & **operator=** (promise &&__rhs) noexcept
- promise & **operator=** (promise &&__rhs) noexcept
- void **set_exception** (exception_ptr __p)
- void **set_exception** (exception_ptr __p)
- void **set_exception_at_thread_exit** (exception_ptr __p)
- void **set_exception_at_thread_exit** (exception_ptr __p)
- void **set_value** (_Res &&__r)
- void **set_value** (_Res &__r)
- void **set_value** (const _Res &__r)
- void **set_value_at_thread_exit** (_Res &&__r)
- void **set_value_at_thread_exit** (_Res &__r)
- void **set_value_at_thread_exit** (const _Res &__r)
- void **swap** (promise &__rhs) noexcept
- void **swap** (promise &__rhs) noexcept

Friends

- struct **_State::_Setter**
- template<typename, typename>
  struct **_State::_Setter**

5.801.1 Detailed Description

template<typename _Res>
class std::promise< _Res & >

Partial specialization for promise<R&>
The documentation for this class was generated from the following file:

- future

5.802 std::promise< void > Class Reference

`#include <future>`

Public Member Functions

- **promise** (allocator_arg_t, const _Allocator &, promise &&__rhs)
- template<typename _Allocator>
  promise (allocator_arg_t, const _Allocator &, promise &&__rhs)
- **promise** (allocator_arg_t, const _Allocator &__a)
- template<typename _Allocator>
  promise (allocator_arg_t, const _Allocator &__a)
- **promise** (const promise &)=delete
- **promise** (const promise &)=delete
- **promise** (promise &&__rhs) noexcept
- **promise** (promise &&__rhs) noexcept
- future< void > **get_future** ()
- future< void > **get_future** ()
- promise & **operator=** (const promise &)=delete
- promise & **operator=** (const promise &)=delete
- promise & **operator=** (promise &&__rhs) noexcept
- promise & **operator=** (promise &&__rhs) noexcept
- void **set_exception** (exception_ptr __p)
- void **set_exception** (exception_ptr __p)
- void **set_exception_at_thread_exit** (exception_ptr __p)
- void **set_exception_at_thread_exit** (exception_ptr __p)
- void **set_value** ()
- void **set_value** (const void &__r)
- void **set_value** (void &&__r)
- void **set_value_at_thread_exit** ()
- void **set_value_at_thread_exit** (const void &__r)
- void **set_value_at_thread_exit** (void &&__r)
- void **swap** (promise &__rhs) noexcept
- void **swap** (promise &__rhs) noexcept

Friends

- struct **_State::_Setter**
- template<typename, typename>
  struct **_State::_Setter**

5.802.1 Detailed Description

Explicit specialization for promise<void>
The documentation for this class was generated from the following file:

- future

5.803 std::experimental::fundamentals_v2::propagate_const< _Tp > Class Template Reference

`#include <propagate_const>`

Public Types

- using **element_type**

Public Member Functions

- template<typename _Up, typename enable_if< __and_< is_constructible< _Tp, _Up && >, is_convertible< _Up &&, _Tp >, __not_< __is_propagate_const< typename decay< _Up >::type > > >::value, bool >::type = true>
  constexpr **propagate_const** (_Up &&__u)
- template<typename _Up, typename enable_if< __and_< is_constructible< _Tp, _Up && >, __not_< is_convertible< _Up &&, _Tp > >, __not_< __is_propagate_const< typename decay< _Up >::type > > >::value, bool >::type = false>
  constexpr **propagate_const** (_Up &&__u)
- **propagate_const** (const propagate_const &__p)=delete
- constexpr **propagate_const** (propagate_const &&__p)=default
- template<typename _Up, typename enable_if< __and_< is_constructible< _Tp, _Up && >, is_convertible< _Up &&, _Tp > >::value, bool >::type = true>
  constexpr **propagate_const** (propagate_const< _Up > &&__pu)
- template<typename _Up, typename enable_if< __and_< is_constructible< _Tp, _Up && >, __not_< is_convertible< _Up &&, _Tp > > >::value, bool >::type = false>
  constexpr **propagate_const** (propagate_const< _Up > &&__pu)
- constexpr element_type * **get** ()
- constexpr const element_type * **get** () const
- constexpr **operator bool** () const
- constexpr element_type & **operator*** ()
- constexpr const element_type & **operator*** () const
- constexpr element_type * **operator->** ()
- constexpr const element_type * **operator->** () const
- template<typename _Up, typename = typename enable_if<__and_<is_convertible<_Up&&, _Tp>, __not_<__is_propagate_const< typename decay<_Up>::type>> >::value>::type>
  constexpr propagate_const & **operator=** (_Up &&__u)
- propagate_const & **operator=** (const propagate_const &__p)=delete
- constexpr propagate_const & **operator=** (propagate_const &&__p)=default
- template<typename _Up, typename = typename enable_if<is_convertible<_Up&&, _Tp>::value>::type>
  constexpr propagate_const & **operator=** (propagate_const< _Up > &&__pu)
- constexpr void **swap** (propagate_const &__pt) noexcept(__is_nothrow_swappable< _Tp >::value)

Friends

- template<typename _Up>
  constexpr const _Up & **get_underlying** (const propagate_const< _Up > &__pt) noexcept
- template<typename _Up>
  constexpr _Up & **get_underlying** (propagate_const< _Up > &__pt) noexcept

5.803.1 Detailed Description

template<typename _Tp>
class std::experimental::fundamentals_v2::propagate_const< _Tp >

Const-propagating wrapper.
The documentation for this class was generated from the following file:

- propagate_const

5.804 __gnu_pbds::quadratic_probe_fn< Size_Type > Class Template Reference

```
#include <hash_policy.hpp>
```

Public Types

- typedef Size_Type **size_type**

Public Member Functions

- void **swap** (quadratic_probe_fn< Size_Type > &other)

Protected Member Functions

- size_type operator() (size_type i) const

5.804.1 Detailed Description

template<typename Size_Type = std::size_t>
class __gnu_pbds::quadratic_probe_fn< Size_Type >

A probe sequence policy using square increments.

5.804.2 Member Function Documentation

operator()()

```
template<typename Size_Type>
quadratic_probe_fn< Size_Type >::size_type __gnu_pbds::quadratic_probe_fn< Size_Type >::operator()
(
            size_type i) const  [inline], [protected]
```

Returns the i-th offset from the hash value.
The documentation for this class was generated from the following file:

- hash_policy.hpp

5.805 std::queue< _Tp, _Sequence > Class Template Reference

```
#include <stl_queue.h>
```

Public Types

- typedef _Sequence::const_reference **const_reference**
- typedef _Sequence **container_type**
- typedef _Sequence::reference **reference**
- typedef _Sequence::size_type **size_type**
- typedef _Sequence::value_type **value_type**

Public Member Functions

- template<typename _Seq = _Sequence, typename _Requires = typename enable_if<is_default_constructible<_Seq>::value>::type>
  queue ()
- **queue** (_Sequence &&__c)
- template<typename _Alloc, typename _Requires = _Uses<_Alloc>>
  queue (_Sequence &&__c, const _Alloc &__a)
- template<typename _Alloc, typename _Requires = _Uses<_Alloc>>
  queue (const _Alloc &__a)
- **queue** (const _Sequence &__c)
- template<typename _Alloc, typename _Requires = _Uses<_Alloc>>
  queue (const _Sequence &__c, const _Alloc &__a)
- template<typename _Alloc, typename _Requires = _Uses<_Alloc>>
  queue (const queue &__q, const _Alloc &__a)
- template<typename _Alloc, typename _Requires = _Uses<_Alloc>>
  queue (queue &&__q, const _Alloc &__a)
- reference back ()
- const_reference back () const
- template<typename... _Args>
  decltype(auto) **emplace** (_Args &&... __args)
- bool empty () const
- reference front ()
- const_reference front () const
- void pop ()
- void push (const value_type &__x)
- void **push** (value_type &&__x)
- size_type size () const
- void **swap** (queue &__q) noexcept(__is_nothrow_swappable< _Sequence >::value)

Protected Attributes

- _Sequence c

Friends

- template<typename _Tp1, typename _Seq1>
  bool **operator**< (const queue< _Tp1, _Seq1 > &, const queue< _Tp1, _Seq1 > &)
- template<typename _Tp1, three_way_comparable _Seq1>
  compare_three_way_result_t< _Seq1 > **operator**<**=**> (const queue< _Tp1, _Seq1 > &, const queue< _Tp1, _Seq1 > &)
- template<typename _Tp1, typename _Seq1>
  bool **operator==** (const queue< _Tp1, _Seq1 > &, const queue< _Tp1, _Seq1 > &)

5.805.1 Detailed Description

template<typename _Tp, typename _Sequence = deque<_Tp>>
class std::queue< _Tp, _Sequence >

A standard container giving FIFO behavior.

Template Parameters

_Tp	Type of element.
_Sequence	Type of underlying sequence, defaults to deque<_Tp>.

Meets many of the requirements of a `container`, but does not define anything to do with iterators. Very few of the other standard container interfaces are defined.
This is not a true container, but an *adaptor*. It holds another container, and provides a wrapper interface to that container. The wrapper is what enforces strict first-in-first-out queue behavior.
The second template parameter defines the type of the underlying sequence/container. It defaults to std::deque, but it can be any type that supports `front`, `back`, `push_back`, and `pop_front`, such as std::list or an appropriate user-defined type.
Members not found in *normal* containers are `container_type`, which is a typedef for the second Sequence parameter, and `push` and `pop`, which are standard queue/FIFO operations.

5.805.2 Constructor & Destructor Documentation

queue()

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
template<typename _Seq = _Sequence, typename _Requires = typename enable_if<is_default_constructible<↩
_Seq>::value>::type>
std::queue< _Tp, _Sequence >::queue ()  [inline]
```

Default constructor creates no elements.
References c.

5.805.3 Member Function Documentation

back() [1/2]

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
reference std::queue< _Tp, _Sequence >::back ()  [inline], [nodiscard]
```

Returns a read/write reference to the data at the last element of the queue.
References c.

back() [2/2]

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
const_reference std::queue< _Tp, _Sequence >::back () const  [inline], [nodiscard]
```

Returns a read-only (constant) reference to the data at the last element of the queue.
References c.

empty()

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
bool std::queue< _Tp, _Sequence >::empty () const  [inline], [nodiscard]
```

Returns true if the queue is empty.
References c.

front() [1/2]

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
reference std::queue< _Tp, _Sequence >::front ()  [inline], [nodiscard]
```

Returns a read/write reference to the data at the first element of the queue.
References c.

front() [2/2]

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
const_reference std::queue< _Tp, _Sequence >::front () const  [inline], [nodiscard]
```

Returns a read-only (constant) reference to the data at the first element of the queue.
References c.

pop()

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
void std::queue< _Tp, _Sequence >::pop ()  [inline]
```

Removes first element.
This is a typical queue operation. It shrinks the queue by one. The time complexity of the operation depends on the underlying sequence.
Note that no data is returned, and if the first element's data is needed, it should be retrieved before pop() is called.
References c.

push()

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
void std::queue< _Tp, _Sequence >::push (
            const value_type & __x)  [inline]
```

Add data to the end of the queue.

Parameters

_↩ _x	Data to be added.

This is a typical queue operation. The function creates an element at the end of the queue and assigns the given data to it. The time complexity of the operation depends on the underlying sequence.
References c.

size()

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
size_type std::queue< _Tp, _Sequence >::size () const  [inline], [nodiscard]
```

Returns the number of elements in the queue.
References c.

5.805.4 Member Data Documentation

c

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
_Sequence std::queue< _Tp, _Sequence >::c  [protected]
```

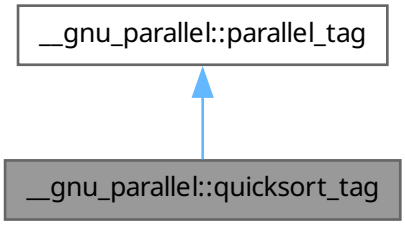
c is the underlying container.
Referenced by queue(), back(), back(), empty(), front(), front(), std::operator<(), std::operator==(), pop(), push(), and size().

The documentation for this class was generated from the following file:

- stl_queue.h

5.806 __gnu_parallel::quicksort_tag Struct Reference

`#include <tags.h>`
Inheritance diagram for __gnu_parallel::quicksort_tag:



Public Member Functions

- **quicksort_tag** (_ThreadIndex __num_threads)
- _ThreadIndex __get_num_threads ()
- void set_num_threads (_ThreadIndex __num_threads)

5.806.1 Detailed Description

Forces parallel sorting using unbalanced quicksort at compile time.

5.806.2 Member Function Documentation

__get_num_threads()

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ()  [inline], [inherited]
```
Find out desired number of threads.

Returns

Desired number of threads.

Referenced by __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), and __gnu_parallel::__parallel_sort(

set_num_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
            _ThreadIndex __num_threads)  [inline], [inherited]
```
Set the desired number of threads.

Parameters

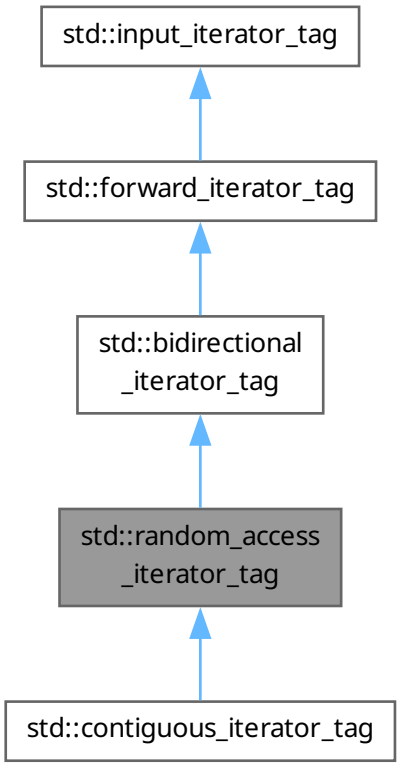
__num_threads	Desired number of threads.

The documentation for this struct was generated from the following file:

- tags.h

5.807 std::random_access_iterator_tag Struct Reference

`#include <stl_iterator_base_types.h>`
Inheritance diagram for std::random_access_iterator_tag:



5.807.1 Detailed Description

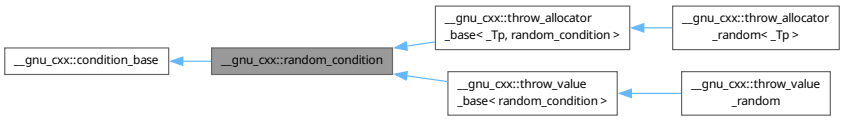
Random-access iterators support a superset of bidirectional iterator operations.
The documentation for this struct was generated from the following file:

- stl_iterator_base_types.h

5.808 __gnu_cxx::random_condition Struct Reference

`#include <throw_allocator.h>`

Inheritance diagram for __gnu_cxx::random_condition:



Classes

- struct always_adjustor
- struct group_adjustor
- struct never_adjustor

Public Member Functions

- void **seed** (unsigned long __s)

Static Public Member Functions

- static void **set_probability** (double __p)
- static void **throw_conditionally** ()

5.808.1 Detailed Description

Base class for random probability control and throw.
The documentation for this struct was generated from the following file:

- throw_allocator.h

5.809 std::random_device Class Reference

`#include <random>`

Public Types

- typedef unsigned int result_type

Public Member Functions

- **random_device** (const random_device &)=delete
- **random_device** (const std::string &__token)
- double **entropy** () const noexcept
- result_type **operator()** ()
- void **operator=** (const random_device &)=delete

Static Public Member Functions

- static constexpr result_type **max** ()
- static constexpr result_type **min** ()

5.809.1 Detailed Description

A standard interface to a platform-specific non-deterministic random number generator (if any are available).

Since

C++11

5.809.2 Member Typedef Documentation

result_type

```
typedef unsigned int std::random_device::result_type
```

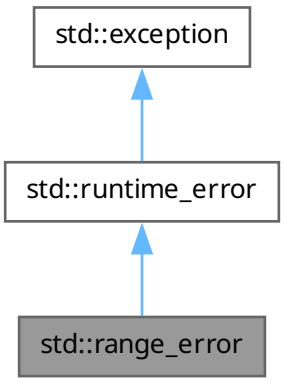
The type of the generated random value.
The documentation for this class was generated from the following file:

- random.h

5.810 std::range_error Class Reference

```
#include <stdexcept>
```

Inheritance diagram for std::range_error:



Public Member Functions

- **range_error** (const char *)
- **range_error** (const range_error &)=default
- **range_error** (const string &__arg)
- **range_error** (range_error &&)=default
- range_error & **operator=** (const range_error &)=default
- range_error & **operator=** (range_error &&)=default
- virtual const char * what () const noexcept

5.810.1 Detailed Description

Thrown to indicate range errors in internal computations.

5.810.2 Member Function Documentation

what()

```
virtual const char * std::runtime_error::what () const  [virtual], [noexcept], [inherited]
```

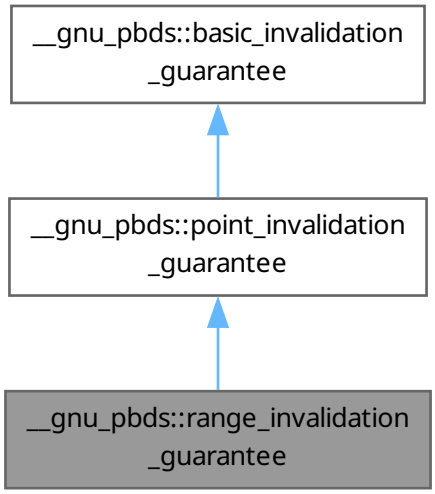
Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).
Reimplemented from std::exception.
Reimplemented in std::experimental::filesystem::v1::filesystem_error, and std::filesystem::filesystem_error.
The documentation for this class was generated from the following file:

- stdexcept

5.811 __gnu_pbds::range_invalidation_guarantee Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for __gnu_pbds::range_invalidation_guarantee:



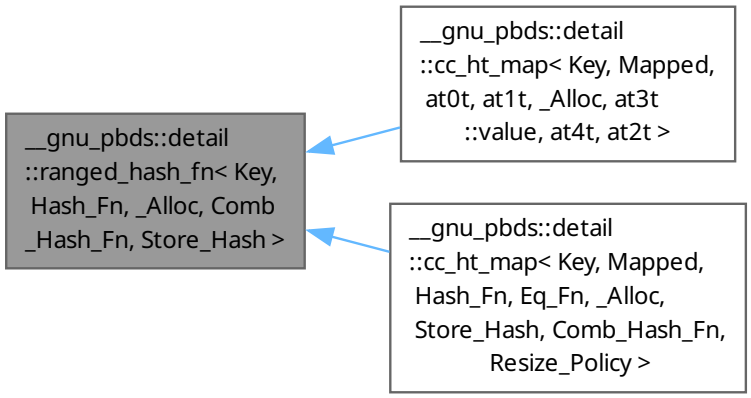
5.811.1 Detailed Description

Signifies an invalidation guarantee that includes all those of its base, and additionally, that any range-type iterator (including the returns of begin() and end()) is in the correct relative positions to other range-type iterators as long as its corresponding entry has not be erased, regardless of modifications to the container object.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

5.812 __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash >:



5.812.1 Detailed Description

template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Hash_Fn, bool Store_Hash> class __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash >

Primary template.
The documentation for this class was generated from the following file:

- ranged_hash_fn.hpp

5.813 __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, false > Class Template Reference

`#include <ranged_hash_fn.hpp>`

Protected Types

- typedef Comb_Hash_Fn **comb_hash_fn_base**
- typedef Hash_Fn **hash_fn_base**
- typedef rebind_traits< _Alloc, Key >::const_reference **key_const_reference**
- typedef _Alloc::size_type **size_type**

Protected Member Functions

- **ranged_hash_fn** (size_type)
- **ranged_hash_fn** (size_type, const Hash_Fn &)
- **ranged_hash_fn** (size_type, const Hash_Fn &, const Comb_Hash_Fn &)
- void **notify_resized** (size_type)
- size_type **operator()** (key_const_reference) const
- void **swap** (ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, false > &)

5.813.1 Detailed Description

template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Hash_Fn>
class __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, false >

Specialization 1 The client supplies a hash function and a ranged hash function, and requests that hash values not be stored.
The documentation for this class was generated from the following file:

- ranged_hash_fn.hpp

5.814 __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true > Class Template Reference

`#include <ranged_hash_fn.hpp>`

Protected Types

- typedef Comb_Hash_Fn **comb_hash_fn_base**
- typedef std::pair< size_type, size_type > **comp_hash**
- typedef Hash_Fn **hash_fn_base**
- typedef rebind_traits< _Alloc, Key >::const_reference **key_const_reference**
- typedef _Alloc::size_type **size_type**

Protected Member Functions

- **ranged_hash_fn** (size_type)
- **ranged_hash_fn** (size_type, const Hash_Fn &)
- **ranged_hash_fn** (size_type, const Hash_Fn &, const Comb_Hash_Fn &)
- void **notify_resized** (size_type)
- comp_hash **operator()** (key_const_reference) const
- comp_hash **operator()** (key_const_reference, size_type) const
- void **swap** (ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true > &)

5.814.1 Detailed Description

template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Hash_Fn>
class __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true >

Specialization 2 The client supplies a hash function and a ranged hash function, and requests that hash values be stored.
The documentation for this class was generated from the following file:

- ranged_hash_fn.hpp

5.815 __gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, false > Class Template Reference

`#include <ranged_hash_fn.hpp>`

Protected Types

- typedef Comb_Hash_Fn **comb_hash_fn_base**
- typedef _Alloc::size_type **size_type**

Protected Member Functions

- **ranged_hash_fn** (size_type)
- **ranged_hash_fn** (size_type, const Comb_Hash_Fn &)
- **ranged_hash_fn** (size_type, const null_type &, const Comb_Hash_Fn &)
- void **swap** (ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, false > &)

5.815.1 Detailed Description

template<typename Key, typename _Alloc, typename Comb_Hash_Fn>
class __gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, false >

Specialization 3 The client does not supply a hash function (by specifying null_type as the Hash_Fn parameter), and requests that hash values not be stored.
The documentation for this class was generated from the following file:

- ranged_hash_fn.hpp

5.816 __gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, true > Class Template Reference

`#include <ranged_hash_fn.hpp>`

Protected Types

- typedef Comb_Hash_Fn **comb_hash_fn_base**
- typedef _Alloc::size_type **size_type**

Protected Member Functions

- **ranged_hash_fn** (size_type)
- **ranged_hash_fn** (size_type, const Comb_Hash_Fn &)
- **ranged_hash_fn** (size_type, const null_type &, const Comb_Hash_Fn &)
- void **swap** (ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, true > &)

5.816.1 Detailed Description

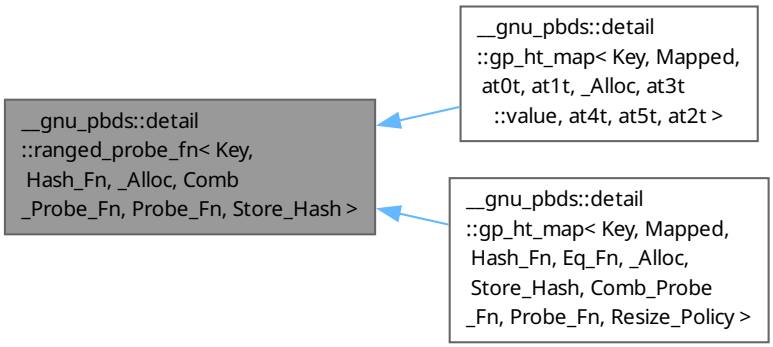
template<typename Key, typename _Alloc, typename Comb_Hash_Fn>
class __gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, true >

Specialization 4 The client does not supply a hash function (by specifying null_type as the Hash_Fn parameter), and requests that hash values be stored.
The documentation for this class was generated from the following file:

- ranged_hash_fn.hpp

5.817 __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_Hash > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_Hash >:



5.817.1 Detailed Description

template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Probe_Fn, typename Probe_Fn, bool Store_Hash>
class __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_Hash >

Primary template.
The documentation for this class was generated from the following file:

- ranged_probe_fn.hpp

5.818 __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, false > Class Template Reference

`#include <ranged_probe_fn.hpp>`

Protected Types

- typedef Comb_Probe_Fn **comb_probe_fn_base**
- typedef Hash_Fn **hash_fn_base**
- typedef rebind_traits< _Alloc, Key >::const_reference **key_const_reference**
- typedef Probe_Fn **probe_fn_base**
- typedef _Alloc::size_type **size_type**

Protected Member Functions

- **ranged_probe_fn** (size_type)
- **ranged_probe_fn** (size_type, const Hash_Fn &)
- **ranged_probe_fn** (size_type, const Hash_Fn &, const Comb_Probe_Fn &)
- **ranged_probe_fn** (size_type, const Hash_Fn &, const Comb_Probe_Fn &, const Probe_Fn &)

- void **notify_resized** (size_type)
- size_type **operator()** (key_const_reference) const
- size_type **operator()** (key_const_reference, size_type, size_type) const
- void **swap** (ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, false > &)

5.818.1 Detailed Description

template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Probe_Fn, typename Probe_Fn>
class __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, false >

Specialization 1
The client supplies a probe function and a ranged probe function, and requests that hash values not be stored.
The documentation for this class was generated from the following file:

- ranged_probe_fn.hpp

5.819 __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true > Class Template Reference

`#include <ranged_probe_fn.hpp>`

Protected Types

- typedef Comb_Probe_Fn **comb_probe_fn_base**
- typedef std::pair< size_type, size_type > **comp_hash**
- typedef Hash_Fn **hash_fn_base**
- typedef rebind_traits< _Alloc, Key >::const_reference **key_const_reference**
- typedef Probe_Fn **probe_fn_base**
- typedef _Alloc::size_type **size_type**

Protected Member Functions

- **ranged_probe_fn** (size_type)
- **ranged_probe_fn** (size_type, const Hash_Fn &)
- **ranged_probe_fn** (size_type, const Hash_Fn &, const Comb_Probe_Fn &)
- **ranged_probe_fn** (size_type, const Hash_Fn &, const Comb_Probe_Fn &, const Probe_Fn &)
- void **notify_resized** (size_type)
- comp_hash **operator()** (key_const_reference) const
- size_type **operator()** (key_const_reference, size_type) const
- size_type **operator()** (key_const_reference, size_type, size_type) const
- void **swap** (ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true > &)

5.819.1 Detailed Description

template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Probe_Fn, typename Probe_Fn>
class __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true >

Specialization 2- The client supplies a probe function and a ranged probe function, and requests that hash values not be stored.
The documentation for this class was generated from the following file:

- ranged_probe_fn.hpp

5.820 __gnu_pbds::detail::ranged_probe_fn< Key, null_type, _Alloc, Comb_Probe_Fn, null_type, false > Class Template Reference

`#include <ranged_probe_fn.hpp>`

Protected Types

- typedef Comb_Probe_Fn **comb_probe_fn_base**
- typedef rebind_traits< _Alloc, Key >::const_reference **key_const_reference**
- typedef _Alloc::size_type **size_type**

Protected Member Functions

- **ranged_probe_fn** (size_type size)
- **ranged_probe_fn** (size_type, const Comb_Probe_Fn &r_comb_probe_fn)
- **ranged_probe_fn** (size_type, const null_type &, const Comb_Probe_Fn &r_comb_probe_fn, const null_type &)
- void **swap** (ranged_probe_fn &other)

5.820.1 Detailed Description

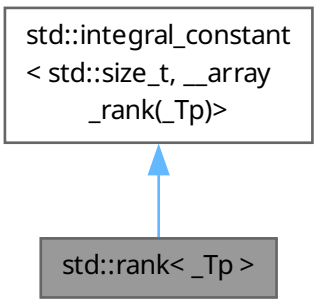
template<typename Key, typename _Alloc, typename Comb_Probe_Fn>
class __gnu_pbds::detail::ranged_probe_fn< Key, null_type, _Alloc, Comb_Probe_Fn, null_type, false >

Specialization 3 and 4 The client does not supply a hash function or probe function, and requests that hash values not be stored.
The documentation for this class was generated from the following file:

- ranged_probe_fn.hpp

5.821 std::rank< _Tp > Struct Template Reference

`#include <type_traits>`
Inheritance diagram for std::rank< _Tp >:



Public Types

- using **type**
- using **value_type**

Public Member Functions

- constexpr **operator value_type** () const noexcept

Static Public Attributes

- static constexpr std::size_t **value**

5.821.1 Detailed Description

template<**typename _Tp**>
struct std::rank< **_Tp** >

rank
The documentation for this struct was generated from the following file:

- type_traits

5.822 std::ratio< _Num, _Den > Struct Template Reference

`#include <ratio>`

Public Types

- typedef ratio< num, den > **type**

Static Public Attributes

- static constexpr intmax_t **den**
- static constexpr intmax_t **num**

5.822.1 Detailed Description

template<**intmax_t _Num, intmax_t _Den = 1**>
struct std::ratio< **_Num, _Den** >

Provides compile-time rational arithmetic.
This class template represents any finite rational number with a numerator and denominator representable by compile-time constants of type intmax_t. The ratio is simplified when instantiated.
For example:

```
std::ratio<7,-21>::num == -1;
std::ratio<7,-21>::den == 3;
```

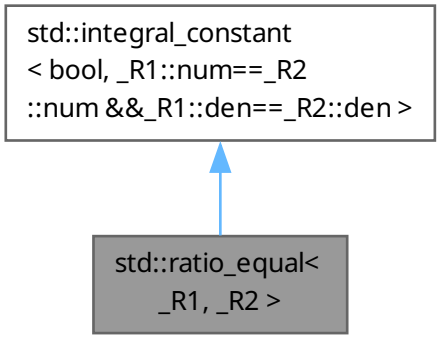
The documentation for this struct was generated from the following file:

- ratio

5.823 std::ratio_equal< _R1, _R2 > Struct Template Reference

`#include <ratio>`

Inheritance diagram for std::ratio_equal< _R1, _R2 >:



Public Types

- using **type**
- using **value_type**

Public Member Functions

- constexpr **operator value_type** () const noexcept

Static Public Attributes

- static constexpr bool **value**

5.823.1 Detailed Description

template<typename _R1, typename _R2>
struct std::ratio_equal< _R1, _R2 >

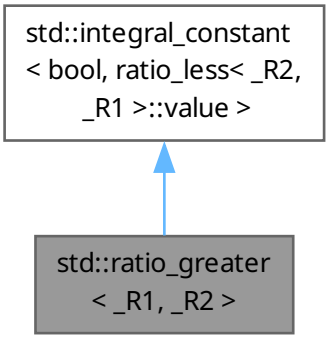
ratio_equal
The documentation for this struct was generated from the following file:

- ratio

5.824 std::ratio_greater< _R1, _R2 > Struct Template Reference

`#include <ratio>`

Inheritance diagram for std::ratio_greater< _R1, _R2 >:



Public Types

- using **type**
- using **value_type**

Public Member Functions

- constexpr **operator value_type** () const noexcept

Static Public Attributes

- static constexpr bool **value**

5.824.1 Detailed Description

template<**typename _R1, typename _R2**>
struct std::ratio_greater< **_R1, _R2** >

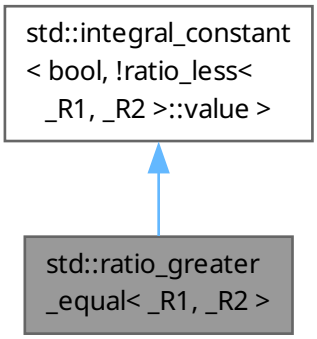
ratio_greater
The documentation for this struct was generated from the following file:

- ratio

5.825 std::ratio_greater_equal< _R1, _R2 > Struct Template Reference

`#include <ratio>`

Inheritance diagram for std::ratio_greater_equal< _R1, _R2 >:



Public Types

- using **type**
- using **value_type**

Public Member Functions

- constexpr **operator value_type** () const noexcept

Static Public Attributes

- static constexpr bool **value**

5.825.1 Detailed Description

template<typename _R1, typename _R2>
struct std::ratio_greater_equal< _R1, _R2 >

ratio_greater_equal
The documentation for this struct was generated from the following file:

- ratio

5.826 std::ratio_less< _R1, _R2 > Struct Template Reference

`#include <ratio>`

5.826.1 Detailed Description

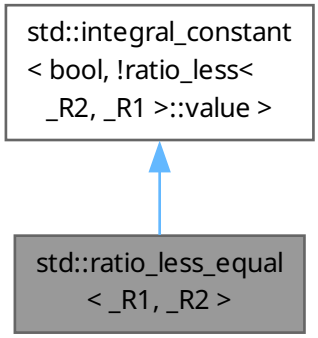
template<typename _R1, typename _R2>
struct std::ratio_less< _R1, _R2 >

ratio_less
The documentation for this struct was generated from the following file:

- ratio

5.827 std::ratio_less_equal< _R1, _R2 > Struct Template Reference

`#include <ratio>`
Inheritance diagram for std::ratio_less_equal< _R1, _R2 >:



Public Types

- using **type**
- using **value_type**

Public Member Functions

- constexpr **operator value_type** () const noexcept

Static Public Attributes

- static constexpr bool **value**

5.827.1 Detailed Description

template<typename _R1, typename _R2>
struct std::ratio_less_equal< _R1, _R2 >

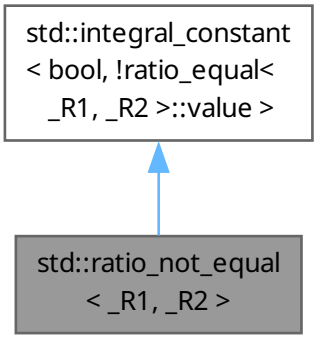
ratio_less_equal
The documentation for this struct was generated from the following file:

- ratio

5.828 std::ratio_not_equal< _R1, _R2 > Struct Template Reference

`#include <ratio>`

Inheritance diagram for std::ratio_not_equal< _R1, _R2 >:



Public Types

- using **type**
- using **value_type**

Public Member Functions

- constexpr **operator value_type** () const noexcept

Static Public Attributes

- static constexpr bool **value**

5.828.1 Detailed Description

template<**typename _R1, typename _R2**>
struct std::ratio_not_equal< **_R1, _R2** >

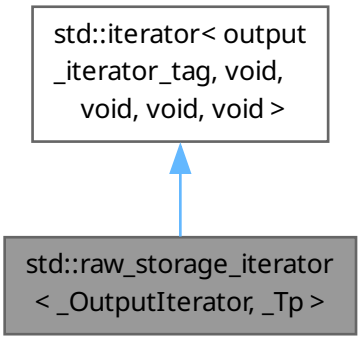
ratio_not_equal
The documentation for this struct was generated from the following file:

- ratio

5.829 std::raw_storage_iterator< _OutputIterator, _Tp > Class Template Reference

`#include <stl_raw_storage_iter.h>`

Inheritance diagram for std::raw_storage_iterator< _OutputIterator, _Tp >:



Public Types

- typedef void difference_type
- typedef output_iterator_tag iterator_category
- typedef void pointer
- typedef void reference
- typedef void value_type

Public Member Functions

- **raw_storage_iterator** (_OutputIterator __x)
- _OutputIterator **base** () const
- raw_storage_iterator & **operator*** ()
- raw_storage_iterator & **operator++** ()
- raw_storage_iterator **operator++** (int)
- raw_storage_iterator & **operator=** (_Tp &&__element)
- raw_storage_iterator & **operator=** (const _Tp &__element)

Protected Attributes

- _OutputIterator **_M_iter**

5.829.1 Detailed Description

template<class _OutputIterator, class _Tp>
class std::raw_storage_iterator< _OutputIterator, _Tp >

This iterator class lets algorithms store their results into uninitialized memory.

5.829.2 Member Typedef Documentation

difference_type

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::difference_type  [inherited]
```

Distance between iterators is represented as this type.

iterator_category

typedef output_iterator_tag std::iterator< output_iterator_tag, void, void, void, void >::iterator_category [inherited]
One of the tag types.

pointer

typedef void std::iterator< output_iterator_tag, void, void, void, void >::pointer [inherited]
This type represents a pointer-to-value_type.

reference

typedef void std::iterator< output_iterator_tag, void, void, void, void >::reference [inherited]
This type represents a reference-to-value_type.

value_type

typedef void std::iterator< output_iterator_tag, void, void, void, void >::value_type [inherited]
The type "pointed to" by the iterator.
The documentation for this class was generated from the following file:

- stl_raw_storage_iter.h

5.830 __gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc > Struct Template Reference

#include <rb_tree>

Public Types

- template<typename _Iter>
  using **__same_value_type**
- typedef std::_Rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc > **_Base**
- typedef _Base::allocator_type **allocator_type**
- typedef _Node_traits::_Const_iterator **const_iterator**
- typedef const value_type * **const_pointer**
- typedef const value_type & **const_reference**
- typedef std::reverse_iterator< const_iterator > **const_reverse_iterator**
- typedef ptrdiff_t **difference_type**
- typedef _Node_traits::_Iterator **iterator**
- typedef _Key **key_type**
- typedef value_type * **pointer**
- typedef value_type & **reference**
- typedef std::reverse_iterator< iterator > **reverse_iterator**
- typedef size_t **size_type**
- typedef _Val **value_type**

Public Member Functions

- **rb_tree** (const _Compare &__comp=_Compare(), const allocator_type &__a=allocator_type())
- **__attribute** ((__abi_tag__("cxx11"))) iterator erase(const_iterator __first
- **__attribute** ((__abi_tag__("cxx11"))) iterator erase(const_iterator __position)
- **__attribute** ((__abi_tag__("cxx11"))) iterator erase(iterator __position)

- bool **__rb_verify** () const
- template<typename _Iterator>
  void **_M_assign_equal** (_Iterator, _Iterator)
- template<typename _Iterator>
  void **_M_assign_unique** (_Iterator, _Iterator)
- template<typename _Kt, typename _Req = __has_is_transparent_t<_Compare, _Kt>>
  size_type **_M_count_tr** (const _Kt &__k) const
- template<typename... _Args>
  iterator **_M_emplace_equal** (_Args &&... __args)
- template<typename... _Args>
  auto **_M_emplace_equal** (_Args &&... __args) -> iterator
- template<typename... _Args>
  iterator **_M_emplace_hint_equal** (const_iterator __pos, _Args &&... __args)
- template<typename... _Args>
  auto **_M_emplace_hint_equal** (const_iterator __pos, _Args &&... __args) -> iterator
- template<typename... _Args>
  iterator **_M_emplace_hint_unique** (const_iterator __pos, _Args &&... __args)
- template<typename... _Args>
  auto **_M_emplace_hint_unique** (const_iterator __pos, _Args &&... __args) -> iterator
- template<typename... _Args>
  pair< iterator, bool > **_M_emplace_unique** (_Args &&... __args)
- template<typename... _Args>
  auto **_M_emplace_unique** (_Args &&... __args) -> pair< iterator, bool >
- template<typename _Kt, typename _Req = __has_is_transparent_t<_Compare, _Kt>>
  pair< iterator, iterator > **_M_equal_range_tr** (const _Kt &__k)
- template<typename _Kt, typename _Req = __has_is_transparent_t<_Compare, _Kt>>
  pair< const_iterator, const_iterator > **_M_equal_range_tr** (const _Kt &__k) const
- size_type **_M_erase_unique** (const key_type &__x)
- template<typename _Kt, typename _Req = __has_is_transparent_t<_Compare, _Kt>>
  iterator **_M_find_tr** (const _Kt &__k)
- template<typename _Kt, typename _Req = __has_is_transparent_t<_Compare, _Kt>>
  const_iterator **_M_find_tr** (const _Kt &__k) const
- pair< _Base_ptr, _Base_ptr > **_M_get_insert_equal_pos** (const key_type &__k)
- pair< _Base_ptr, _Base_ptr > **_M_get_insert_hint_equal_pos** (const_iterator __pos, const key_type &__k)
- pair< _Base_ptr, _Base_ptr > **_M_get_insert_hint_unique_pos** (const_iterator __pos, const key_type &__k)
- pair< _Base_ptr, _Base_ptr > **_M_get_insert_unique_pos** (const key_type &__k)
- const _Node_allocator & **_M_get_Node_allocator** () const noexcept
- _Node_allocator & **_M_get_Node_allocator** () noexcept
- template<typename _Arg>
  iterator **_M_insert_equal** (_Arg &&__x)
- template<typename _Arg>
  iterator **_M_insert_equal_** (const_iterator __pos, _Arg &&__x)
- template<typename _Arg, typename _NodeGen>
  iterator **_M_insert_equal_** (const_iterator __pos, _Arg &&__x, _NodeGen &)
- template<typename _InputIterator>
  __enable_if_t< __same_value_type< _InputIterator >::value > **_M_insert_range_equal** (_InputIterator __first, _InputIterator __last)
- template<typename _InputIterator>
  __enable_if_t<!__same_value_type< _InputIterator >::value > **_M_insert_range_equal** (_InputIterator __first, _InputIterator __last)
- template<typename _InputIterator>
  __enable_if_t< __same_value_type< _InputIterator >::value > **_M_insert_range_unique** (_InputIterator __↩
  first, _InputIterator __last)

- template<typename _InputIterator>
  __enable_if_t<!__same_value_type< _InputIterator >::value > **_M_insert_range_unique** (_InputIterator __first, _InputIterator __last)
- template<typename _Arg>
  pair< typename _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc >::iterator, bool > **_M_insert_unique** (_Arg &&__v)
- template<typename _Arg>
  pair< iterator, bool > **_M_insert_unique** (_Arg &&__x)
- template<typename _Arg>
  iterator **_M_insert_unique_** (const_iterator __pos, _Arg &&__x)
- template<typename _Arg, typename _NodeGen>
  iterator **_M_insert_unique_** (const_iterator __pos, _Arg &&__x, _NodeGen &)
- template<typename _Kt, typename _Req = __has_is_transparent_t<_Compare, _Kt>>
  _Base_ptr **_M_lower_bound_tr** (const _Kt &__k) const
- template<typename _Kt, typename _Req = __has_is_transparent_t<_Compare, _Kt>>
  _Base_ptr **_M_upper_bound_tr** (const _Kt &__k) const
- const_iterator **begin** () const noexcept
- iterator **begin** () noexcept
- void **clear** () noexcept
- size_type **count** (const key_type &__k) const
- bool **empty** () const noexcept
- const_iterator **end** () const noexcept
- iterator **end** () noexcept
- pair< iterator, iterator > **equal_range** (const key_type &__k)
- pair< const_iterator, const_iterator > **equal_range** (const key_type &__k) const
- size_type **erase** (const key_type &__x)
- iterator **find** (const key_type &__k)
- const_iterator **find** (const key_type &__k) const
- allocator_type **get_allocator** () const noexcept
- return **iterator** (__last._M_node)
- _Compare **key_comp** () const
- iterator **lower_bound** (const key_type &__k)
- const_iterator **lower_bound** (const key_type &__k) const
- size_type **max_size** () const noexcept
- const_reverse_iterator **rbegin** () const noexcept
- reverse_iterator **rbegin** () noexcept
- const_reverse_iterator **rend** () const noexcept
- reverse_iterator **rend** () noexcept
- size_type **size** () const noexcept
- void **swap** (_Rb_tree &__t) noexcept(/∗conditional ∗/)
- iterator **upper_bound** (const key_type &__k)
- const_iterator **upper_bound** (const key_type &__k) const

Public Attributes

- const_iterator **__last**

Protected Types

- typedef _Node_traits::_Base_ptr **_Base_ptr**
- typedef _Node_traits::_Header_t **_Header_t**
- typedef _Node_traits::_Node_ptr **_Node_ptr**

Protected Member Functions

- _Base_ptr **_M_begin** () const noexcept
- _Node_ptr **_M_begin_node** () const noexcept
- template<bool _MoveValue, typename _NodeGen>
  _Node_ptr **_M_clone_node** (_Node_ptr __x, _NodeGen &__node_gen)
- template<typename... _Args>
  void **_M_construct_node** (_Node_ptr __node, _Args &&... __args)
- template<typename... _Args>
  _Node_ptr **_M_create_node** (_Args &&... __args)
- void **_M_destroy_node** (_Node_ptr __p) noexcept
- void **_M_drop_node** (_Node_ptr __p) noexcept
- _Base_ptr **_M_end** () const noexcept
- _Node_ptr **_M_get_node** ()
- template<typename _Key1, typename _Key2>
  bool **_M_key_compare** (const _Key1 &__k1, const _Key2 &__k2) const
- _Base_ptr **_M_leftmost** () const noexcept
- _Base_ptr & **_M_leftmost** () noexcept
- void **_M_put_node** (_Node_ptr __p) noexcept
- _Base_ptr **_M_rightmost** () const noexcept
- _Base_ptr & **_M_rightmost** () noexcept
- _Base_ptr **_M_root** () const noexcept
- _Base_ptr & **_M_root** () noexcept

Static Protected Member Functions

- static const _Key & **_S_key** (_Base_ptr __x)
- static const _Key & **_S_key** (_Node_ptr __x)
- static const _Key & **_S_key** (const _Node &__node)
- static _Base_ptr **_S_left** (_Base_ptr __x) noexcept
- static _Node_ptr **_S_left** (_Node_ptr __x)
- static _Base_ptr **_S_right** (_Base_ptr __x) noexcept
- static _Node_ptr **_S_right** (_Node_ptr __x) noexcept

Protected Attributes

- _Rb_tree_impl< _Compare > **_M_impl**

5.830.1 Detailed Description

template<class _Key, class _Value, class _KeyOfValue, class _Compare, class _Alloc = std::allocator<_↩ Value>>
struct __gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc >

This is an SGI extension.

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

The documentation for this struct was generated from the following file:

- rb_tree

5.831 __gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > Class Template Reference

`#include <rb_tree_.hpp>`

Public Types

- typedef _Alloc **allocator_type**
- typedef Cmp_Fn **cmp_fn**
- typedef std::pair< size_type, size_type > **comp_hash**
- typedef base_type::const_iterator **const_iterator**
- typedef base_type::const_pointer **const_pointer**
- typedef base_type::const_reference **const_reference**
- typedef base_type::const_reverse_iterator **const_reverse_iterator**
- typedef rb_tree_tag **container_category**
- typedef _Alloc::difference_type **difference_type**
- typedef base_type::iterator **iterator**
- typedef base_type::key_const_pointer **key_const_pointer**
- typedef base_type::key_const_reference **key_const_reference**
- typedef base_type::key_pointer **key_pointer**
- typedef base_type::key_reference **key_reference**
- typedef base_type::key_type **key_type**
- typedef base_type::mapped_const_pointer **mapped_const_pointer**
- typedef base_type::mapped_const_reference **mapped_const_reference**
- typedef base_type::mapped_pointer **mapped_pointer**
- typedef base_type::mapped_reference **mapped_reference**
- typedef base_type::mapped_type **mapped_type**
- typedef __nothrowcopy::indicator **no_throw_indicator**
- typedef traits_type::node_const_iterator **node_const_iterator**
- typedef traits_type::node_iterator **node_iterator**
- typedef base_type::node_update **node_update**
- typedef base_type::const_iterator **point_const_iterator**
- typedef base_type::point_iterator **point_iterator**
- typedef base_type::pointer **pointer**
- typedef base_type::reference **reference**
- typedef base_type::reverse_iterator **reverse_iterator**
- typedef _Alloc::size_type **size_type**
- typedef integral_constant< int, Store_Hash > **store_extra**
- typedef stored_data< value_type, size_type, Store_Hash > **stored_data_type**
- typedef base_type::value_type **value_type**

Public Member Functions

- **rb_tree_map** (const Cmp_Fn &)
- **rb_tree_map** (const Cmp_Fn &, const node_update &)
- **rb_tree_map** (const rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- template<typename It>
  void **copy_from_range** (It, It)
- bool **empty** () const

- iterator **end** ()
- const_iterator **end** () const
- iterator **erase** (iterator)
- bool **erase** (key_const_reference)
- reverse_iterator **erase** (reverse_iterator)
- template<typename Pred>
  size_type **erase_if** (Pred)
- point_iterator **find** (key_const_reference)
- point_const_iterator **find** (key_const_reference) const
- Cmp_Fn & **get_cmp_fn** ()
- const Cmp_Fn & **get_cmp_fn** () const
- std::pair< point_iterator, bool > **insert** (const_reference)
- void **join** (rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- point_iterator **lower_bound** (key_const_reference)
- point_const_iterator **lower_bound** (key_const_reference) const
- size_type **max_size** () const
- node_iterator node_begin ()
- node_const_iterator node_begin () const
- node_iterator node_end ()
- node_const_iterator node_end () const
- mapped_reference **operator[ ]** (key_const_reference r_key)
- reverse_iterator **rbegin** ()
- const_reverse_iterator **rbegin** () const
- reverse_iterator **rend** ()
- const_reverse_iterator **rend** () const
- size_type **size** () const
- void **split** (key_const_reference, rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- void **swap** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- void **swap** (rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- point_iterator **upper_bound** (key_const_reference)
- point_const_iterator **upper_bound** (key_const_reference) const

Public Attributes

- no_throw_indicator **m_no_throw_copies_indicator**
- no_throw_indicator **m_no_throw_copies_indicator**
- store_extra **m_store_extra_indicator**
- store_extra **m_store_extra_indicator**

Protected Types

- typedef node_alloc_traits::value_type **node**
- typedef node_alloc_traits::allocator_type **node_allocator**
- typedef traits_type::null_node_update_pointer **null_node_update_pointer**
- typedef types_traits< Key, Mapped, _Alloc, false > **traits_base**

Protected Member Functions

- void **actual_erase_node** (node_pointer)
- template<typename Node_Update_>
  void **apply_update** (node_pointer, Node_Update_ *)
- void **apply_update** (node_pointer, null_node_update_pointer)
- std::pair< node_pointer, bool > **erase** (node_pointer)
- node_pointer **get_new_node_for_leaf_insert** (const_reference, false_type)
- node_pointer **get_new_node_for_leaf_insert** (const_reference, true_type)
- void **initialize_min_max** ()
- iterator **insert_imp_empty** (const_reference)
- std::pair< point_iterator, bool > **insert_leaf** (const_reference)
- iterator **insert_leaf_new** (const_reference, node_pointer, bool)
- void **join_finish** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- bool **join_prep** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- size_type **recursive_count** (node_pointer) const
- void **rotate_left** (node_pointer)
- void **rotate_parent** (node_pointer)
- void **rotate_right** (node_pointer)
- void **split_finish** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- bool **split_prep** (key_const_reference, bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _↩ Alloc > &)
- void **update_min_max_for_erased_node** (node_pointer)
- void **update_subtree_size** (node_pointer)
- template<typename Node_Update_>
  void **update_to_top** (node_pointer, Node_Update_ *)
- void **update_to_top** (node_pointer, null_node_update_pointer)
- void **value_swap** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)

Static Protected Member Functions

- static void **clear_imp** (node_pointer)

Protected Attributes

- node_pointer **m_p_head**
- size_type **m_size**

Static Protected Attributes

- static node_allocator **s_node_allocator**

5.831.1 Detailed Description

template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _↩ Alloc>
class __gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >

Red-Black tree.
This implementation uses an idea from the SGI STL (using a *header* node which is needed for efficient iteration).

5.831.2 Member Function Documentation

node_begin() [1/2]

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _↩
Alloc>
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator __gnu_↩
pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin
() [inline], [inherited]
```

Returns a node_iterator corresponding to the node at the root of the tree.

node_begin() [2/2]

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _↩
Alloc>
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __↩
gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_↩
begin () const [inline], [inherited]
```

Returns a const node_iterator corresponding to the node at the root of the tree.

node_end() [1/2]

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _↩
Alloc>
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator __gnu_↩
pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end ()
[inline], [inherited]
```

Returns a node_iterator corresponding to a node just after a leaf of the tree.

node_end() [2/2]

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _↩
Alloc>
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __↩
gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_↩
end () const [inline], [inherited]
```

Returns a const node_iterator corresponding to a node just after a leaf of the tree.

The documentation for this class was generated from the following file:

- rb_tree_.hpp

5.832 __gnu_pbds::detail::rb_tree_node_< Value_Type, Metadata, _Alloc > Struct Template Reference

```
#include <node.hpp>
```

Public Types

- typedef rebind_traits< _Alloc, metadata_type >::const_reference **metadata_const_reference**
- typedef rebind_traits< _Alloc, metadata_type >::reference **metadata_reference**
- typedef Metadata **metadata_type**
- typedef rebind_traits< _Alloc, rb_tree_node_ >::pointer **node_pointer**
- typedef rebind_traits< _Alloc, rb_tree_node_ >::size_type **size_type**
- typedef Value_Type **value_type**

Public Member Functions

- metadata_reference **get_metadata** ()
- metadata_const_reference **get_metadata** () const
- bool **special** () const

Public Attributes

- metadata_type **m_metadata**
- node_pointer **m_p_left**
- node_pointer **m_p_parent**
- node_pointer **m_p_right**
- bool **m_red**
- size_type **m_subtree_size**
- value_type **m_value**

5.832.1 Detailed Description

template<typename Value_Type, class Metadata, typename _Alloc>
struct __gnu_pbds::detail::rb_tree_node_< Value_Type, Metadata, _Alloc >

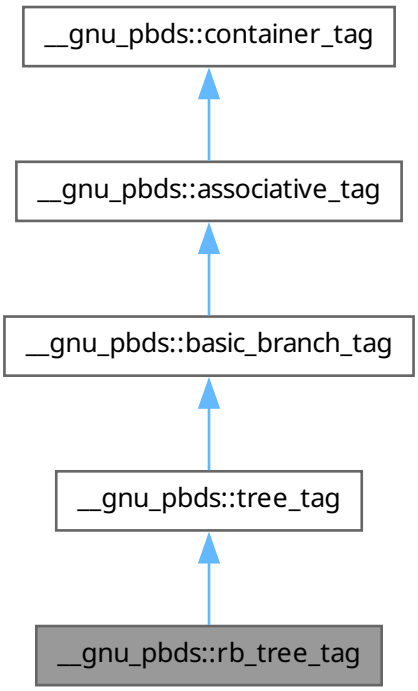
Node for Red-Black trees.
The documentation for this struct was generated from the following file:

- rb_tree_map_/node.hpp

5.833 __gnu_pbds::rb_tree_tag Struct Reference

`#include <tag_and_trait.hpp>`

Inheritance diagram for __gnu_pbds::rb_tree_tag:



5.833.1 Detailed Description

Red-black tree.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

5.834 __gnu_pbds::detail::rc< _Node, _Alloc > Class Template Reference

`#include <rc.hpp>`

Public Types

- typedef entry_const_pointer **const_iterator**
- typedef node_pointer **entry**

Public Member Functions

- **rc** (const rc &)
- const const_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- const const_iterator **end** () const

- void **pop** ()
- void **push** (entry)
- size_type **size** () const
- void **swap** (rc &)
- node_pointer **top** () const

5.834.1 Detailed Description

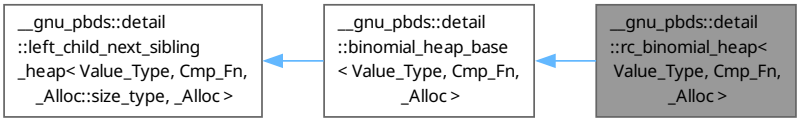
template<typename _Node, typename _Alloc>
class __gnu_pbds::detail::rc< _Node, _Alloc >

Redundant binary counter.
The documentation for this class was generated from the following file:

- rc.hpp

5.835 __gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc > Class Template Reference

`#include <rc_binomial_heap_.hpp>`
Inheritance diagram for __gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc >:



Public Types

- typedef base_type::allocator_type **allocator_type**
- typedef base_type::cmp_fn **cmp_fn**
- typedef base_type::const_iterator **const_iterator**
- typedef base_type::const_pointer **const_pointer**
- typedef base_type::const_reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef base_type::iterator **iterator**
- typedef base_type::point_const_iterator **point_const_iterator**
- typedef base_type::point_iterator **point_iterator**
- typedef base_type::pointer **pointer**
- typedef base_type::reference **reference**
- typedef _Alloc::size_type **size_type**
- typedef Value_Type **value_type**

Public Member Functions

- **rc_binomial_heap** (const Cmp_Fn &)
- **rc_binomial_heap** (const rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc > &)
- iterator **begin** ()
- const_iterator **begin** () const

- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- void **erase** (point_iterator)
- template<typename Pred>
  size_type **erase_if** (Pred)
- Cmp_Fn & **get_cmp_fn** ()
- const Cmp_Fn & **get_cmp_fn** () const
- void **join** (binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > &)
- void **join** (rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc > &)
- size_type **max_size** () const
- void **modify** (point_iterator, const_reference)
- void **pop** ()
- point_iterator **push** (const_reference)
- size_type **size** () const
- template<typename Pred>
  void **split** (Pred, binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > &)
- template<typename Pred>
  void **split** (Pred, rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc > &)
- void **swap** (left_child_next_sibling_heap< Value_Type, Cmp_Fn, _Alloc::size_type, _Alloc > &)
- void **swap** (rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc > &)
- const_reference **top** () const

Protected Types

- typedef base_type::node **node**
- typedef alloc_traits::allocator_type **node_allocator**
- typedef _Alloc::size_type **node_metadata**
- typedef std::pair< node_pointer, node_pointer > **node_pointer_pair**

Protected Member Functions

- void **actual_erase_node** (node_pointer)
- void **bubble_to_top** (node_pointer)
- void **clear_imp** (node_pointer)
- template<typename It>
  void **copy_from_range** (It, It)
- void **find_max** ()
- node_pointer **get_new_node_for_insert** (const_reference)
- node_pointer **prune** (Pred)
- void **swap** (binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > &)
- void **swap_with_parent** (node_pointer, node_pointer)
- void **to_linked_list** ()
- void **value_swap** (left_child_next_sibling_heap &)

Static Protected Member Functions

- static void **make_child_of** (node_pointer, node_pointer)
- static node_pointer **parent** (node_pointer)

Protected Attributes

- node_pointer **m_p_max**
- node_pointer **m_p_root**
- size_type **m_size**

5.835.1 Detailed Description

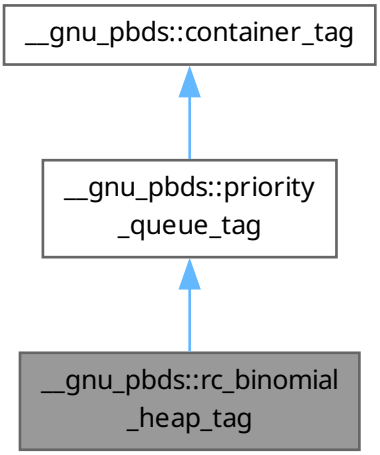
template<typename Value_Type, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc >

Redundant-counter binomial heap.
The documentation for this class was generated from the following file:

- rc_binomial_heap_.hpp

5.836 __gnu_pbds::rc_binomial_heap_tag Struct Reference

`#include <tag_and_trait.hpp>`
Inheritance diagram for __gnu_pbds::rc_binomial_heap_tag:



5.836.1 Detailed Description

Redundant-counter binomial-heap.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

5.837 __gnu_pbds::detail::rebind_traits< _Alloc, T > Struct Template Reference

`#include <types_traits.hpp>`

Public Types

- using **const_reference**
- using **reference**

5.837.1 Detailed Description

template<typename _Alloc, typename T>
struct __gnu_pbds::detail::rebind_traits< _Alloc, T >

Consistent API for accessing allocator-related types.
The documentation for this struct was generated from the following file:

- types_traits.hpp

5.838 std::filesystem::recursive_directory_iterator Class Reference

`#include <filesystem>`

Public Types

- typedef ptrdiff_t **difference_type**
- typedef input_iterator_tag **iterator_category**
- typedef const directory_entry * **pointer**
- typedef const directory_entry & **reference**
- typedef directory_entry **value_type**

Public Member Functions

- **recursive_directory_iterator** (const path &__p)
- **recursive_directory_iterator** (const path &__p, directory_options __options)
- **recursive_directory_iterator** (const path &__p, directory_options __options, error_code &__ec)
- **recursive_directory_iterator** (const path &__p, error_code &__ec)
- **recursive_directory_iterator** (const recursive_directory_iterator &)=default
- **recursive_directory_iterator** (recursive_directory_iterator &&)=default
- int **depth** () const noexcept
- void **disable_recursion_pending** () noexcept
- recursive_directory_iterator & **increment** (error_code &__ec)
- const directory_entry & **operator*** () const noexcept
- recursive_directory_iterator & **operator++** ()
- __directory_iterator_proxy **operator++** (int)
- const directory_entry * **operator->** () const noexcept
- recursive_directory_iterator & **operator=** (const recursive_directory_iterator &__rhs) noexcept
- recursive_directory_iterator & **operator=** (recursive_directory_iterator &&__rhs) noexcept
- bool **operator==** (default_sentinel_t) const noexcept
- directory_options **options** () const noexcept
- void **pop** ()
- void **pop** (error_code &)
- bool **recursion_pending** () const noexcept

Friends

- uintmax_t **filesystem::remove_all** (const path &)
- uintmax_t **filesystem::remove_all** (const path &, error_code &)
- bool **operator==** (const recursive_directory_iterator &__lhs, const recursive_directory_iterator &__rhs) noexcept

5.838.1 Detailed Description

Iterator type for recursively traversing a directory hierarchy.

Since

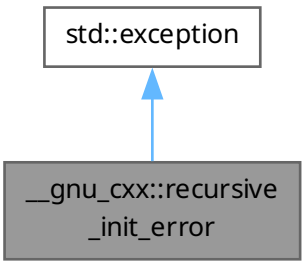
C++17

The documentation for this class was generated from the following file:

- bits/fs_dir.h

5.839 __gnu_cxx::recursive_init_error Class Reference

`#include <cxxabi.h>`
Inheritance diagram for __gnu_cxx::recursive_init_error:



Public Member Functions

- virtual const char * what () const noexcept

5.839.1 Detailed Description

Exception thrown by __cxa_guard_acquire.
C++ 2011 6.7 [stmt.dcl]/4: If control re-enters the declaration recursively while the variable is being initialized, the behavior is undefined.
Since we already have a library function to handle locking, we might as well check for this situation and throw an exception. We use the second byte of the guard variable to remember that we're in the middle of an initialization.

5.839.2 Member Function Documentation

what()

`virtual const char * std::exception::what () const [virtual], [noexcept], [inherited]`
Returns a C-style character string describing the general cause of the current error.
Reimplemented in std::bad_alloc, std::bad_cast, std::bad_exception, std::bad_function_call, std::bad_typeid, std::bad_weak_ptr, std::experimental::filesystem::v1::filesystem_error, std::experimental::fundamentals_v1::bad_any_cast, std::filesystem::filesystem_error, std::future_error, std::logic_error, and std::runtime_error.
The documentation for this class was generated from the following file:

- cxxabi.h

5.840 std::recursive_mutex Class Reference

`#include <mutex>`

Public Types

- typedef __native_type * **native_handle_type**

Public Member Functions

- **recursive_mutex** (const recursive_mutex &)=delete
- void **lock** ()
- native_handle_type **native_handle** () noexcept
- recursive_mutex & **operator=** (const recursive_mutex &)=delete
- bool **try_lock** () noexcept
- void **unlock** ()

5.840.1 Detailed Description

The standard recursive mutex type.
A recursive mutex can be locked more than once by the same thread. Other threads cannot lock the mutex until the owning thread unlocks it as many times as it was locked.

Since

C++11

The documentation for this class was generated from the following file:

- mutex

5.841 std::recursive_timed_mutex Class Reference

`#include <mutex>`

Public Types

- typedef __native_type * **native_handle_type**

Public Member Functions

- **recursive_timed_mutex** (const recursive_timed_mutex &)=delete
- void **lock** ()
- native_handle_type **native_handle** () noexcept
- recursive_timed_mutex & **operator=** (const recursive_timed_mutex &)=delete
- bool **try_lock** () noexcept
- template<class _Rep, class _Period>
  bool **try_lock_for** (const chrono::duration< _Rep, _Period > &__rtime)
- template<class _Clock, class _Duration>
  bool **try_lock_until** (const chrono::time_point< _Clock, _Duration > &__atime)
- void **unlock** ()

Friends

- class **__timed_mutex_impl**< **recursive_timed_mutex** >

5.841.1 Detailed Description

The standard recursive timed mutex type.
A recursive mutex that supports a timeout when trying to acquire the lock. A recursive mutex can be locked more than once by the same thread. Other threads cannot lock the mutex until the owning thread unlocks it as many times as it was locked.

Since

C++11

The documentation for this class was generated from the following file:

- mutex

5.842 std::bitset< _Nb >::reference Class Reference

`#include <bitset>`

Public Member Functions

- **reference** (const reference &)=default
- constexpr reference & **flip** () noexcept
- constexpr **operator bool** () const noexcept
- constexpr reference & **operator=** (bool __x) noexcept
- constexpr reference & **operator=** (const reference &__j) noexcept
- constexpr bool **operator~** () const noexcept

Friends

- class **bitset**

5.842.1 Detailed Description

template<size_t _Nb>
class std::bitset< _Nb >::reference

This encapsulates the concept of a single bit. An instance of this class is a proxy for an actual bit; this way the individual bit operations are done as faster word-size bitwise instructions.
Most users will never need to use this class directly; conversions to and from bool are automatic and should be transparent. Overloaded operators help to preserve the illusion.
(On a typical system, this *bit reference* is 64 times the size of an actual bit. Ha.)
The documentation for this class was generated from the following file:

- bitset

5.843 std::tr2::dynamic_bitset< _WordT, _Alloc >::reference Class Reference

`#include <dynamic_bitset>`

Public Member Functions

- **reference** (dynamic_bitset &__b, size_type __pos) noexcept
- reference & **flip** () noexcept
- **operator bool** () const noexcept
- reference & **operator=** (bool __x) noexcept
- reference & **operator=** (const reference &__j) noexcept
- bool **operator~** () const noexcept

Friends

- class **dynamic_bitset**

5.843.1 Detailed Description

template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
class std::tr2::dynamic_bitset< _WordT, _Alloc >::reference

This encapsulates the concept of a single bit. An instance of this class is a proxy for an actual bit; this way the individual bit operations are done as faster word-size bitwise instructions.
Most users will never need to use this class directly; conversions to and from bool are automatic and should be transparent. Overloaded operators help to preserve the illusion.
(On a typical system, this "bit %reference" is 64 times the size of an actual bit. Ha.)
The documentation for this class was generated from the following file:

- dynamic_bitset

5.844 std::reference_wrapper< _Tp > Class Template Reference

`#include <refwrap.h>`

Public Types

- typedef _Tp **type**

Public Member Functions

- template<typename _Up, typename = __not_same<_Up>, typename = decltype(reference_wrapper::_S_fun(std::declval<_Up>()))>
  constexpr **reference_wrapper** (_Up &&__uref) noexcept(noexcept(reference_wrapper::_S_fun(std::declval< _Up >())))
- **reference_wrapper** (const reference_wrapper &)=default
- constexpr _Tp & **get** () const noexcept
- constexpr **operator _Tp &** () const noexcept
- template<typename... _Args>
  constexpr __invoke_result< _Tp &, _Args... >::type **operator()** (_Args &&... __args) const noexcept(__is_nothrow_invocable< _Tp &, _Args... >::value)
- reference_wrapper & **operator=** (const reference_wrapper &)=default

Related Symbols

(Note that these are not member symbols.)

- template<typename _Tp>
  constexpr reference_wrapper< _Tp > ref (_Tp &__t) noexcept
- template<typename _Tp>
  constexpr reference_wrapper< const _Tp > cref (const _Tp &__t) noexcept
- template<typename _Tp>
  constexpr reference_wrapper< _Tp > ref (reference_wrapper< _Tp > __t) noexcept
- template<typename _Tp>
  constexpr reference_wrapper< const _Tp > cref (reference_wrapper< _Tp > __t) noexcept

5.844.1 Detailed Description

template<typename _Tp>
class std::reference_wrapper< _Tp >

Primary class template for reference_wrapper.

5.844.2 Friends And Related Symbol Documentation

cref() [1/2]

```
template<typename _Tp>
reference_wrapper< const _Tp > cref (
            const _Tp & __t)  [related]
```

Denotes a const reference should be taken to a variable.

cref() [2/2]

```
template<typename _Tp>
reference_wrapper< const _Tp > cref (
            reference_wrapper< _Tp > __t)  [related]
```

std::cref overload to prevent wrapping a reference_wrapper

ref() [1/2]

```
template<typename _Tp>
reference_wrapper< _Tp > ref (
            _Tp & __t)  [related]
```

Denotes a reference should be taken to a variable.

ref() [2/2]

```
template<typename _Tp>
reference_wrapper< _Tp > ref (
            reference_wrapper< _Tp > __t)  [related]
```

std::ref overload to prevent wrapping a reference_wrapper

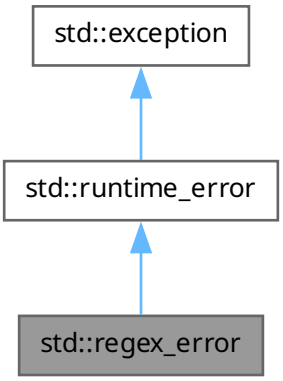
The documentation for this class was generated from the following files:

- type_traits
- refwrap.h

5.845 std::regex_error Class Reference

```
#include <regex>
```

Inheritance diagram for std::regex_error:



Public Member Functions

- regex_error (error_type __ecode)
- regex_constants::error_type code () const noexcept
- virtual const char * what () const noexcept

5.845.1 Detailed Description

A regular expression exception class.
The regular expression library throws objects of this class on error.

Since

C++11

5.845.2 Constructor & Destructor Documentation

regex_error()

```
std::regex_error::regex_error (
            error_type __ecode)  [explicit]
```

Constructs a regex_error object.

Parameters

__ecode	the regex error code.

References regex_error().
Referenced by regex_error().

5.845.3 Member Function Documentation

code()

```
regex_constants::error_type std::regex_error::code () const  [inline], [noexcept]
```

Gets the regex error code.

Returns

the regex error code.

References code().
Referenced by code().

what()

```
virtual const char * std::runtime_error::what () const  [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).
Reimplemented from std::exception.
Reimplemented in std::experimental::filesystem::v1::filesystem_error, and std::filesystem::filesystem_error.
The documentation for this class was generated from the following file:

- regex_error.h

5.846 std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > Class Template Reference

```
#include <regex>
```

Public Types

- typedef std::ptrdiff_t **difference_type**
- typedef std::forward_iterator_tag **iterator_category**
- typedef std::input_iterator_tag **iterator_concept**
- typedef const value_type * **pointer**
- typedef const value_type & **reference**
- typedef basic_regex< _Ch_type, _Rx_traits > **regex_type**
- typedef match_results< _Bi_iter > **value_type**

Public Member Functions

- regex_iterator ()=default
- regex_iterator (_Bi_iter __a, _Bi_iter __b, const regex_type &__re, regex_constants::match_flag_type __m=regex_constants::match_default)
- **regex_iterator** (_Bi_iter, _Bi_iter, const regex_type &&, regex_constants::match_flag_type=regex_constants::match_default)=delete
- regex_iterator (const regex_iterator &)=default
- const value_type & operator* () const noexcept
- regex_iterator & operator++ ()
- regex_iterator operator++ (int)
- const value_type * operator-> () const noexcept
- regex_iterator & operator= (const regex_iterator &)=default
- bool operator== (const regex_iterator &) const noexcept
- bool **operator==** (default_sentinel_t) const noexcept

5.846.1 Detailed Description

template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>>
class std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >

An iterator adaptor that will provide repeated calls of regex_search over a range until no more matches remain.

Since

C++11

5.846.2 Constructor & Destructor Documentation

regex_iterator() [1/3]

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_iterator ()  [default]
```

Provides a singular iterator, useful for indicating one-past-the-end of a range.
Referenced by std::regex_iterator< const char * >::regex_iterator(), and operator==().

regex_iterator() [2/3]

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_iterator (
            _Bi_iter __a,
            _Bi_iter __b,
            const regex_type & __re,
            regex_constants::match_flag_type __m = regex_constants::match_default)  [inline]
```

Constructs a regex_iterator...

Parameters

_↩ _a	[IN] The start of a text range to search.
_↩ _b	[IN] One-past-the-end of the text range to search.
_↩ _re	[IN] The regular expression to match.
_↩ _m	[IN] Policy flags for match rules.

regex_iterator() [3/3]

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_iterator (
            const regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > & )  [default]
```

Copy constructs a regex_iterator.

5.846.3 Member Function Documentation

operator*()

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
const value_type & std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator* () const
[inline], [noexcept]
```

Dereferences a regex_iterator.

operator++() [1/2]

```
template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>
regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > & std::regex_iterator< _Bi_iter, _Ch_type, _Rx↩
_traits >::operator++ ()
```

Increments a regex_iterator.

References std::regex_constants::match_continuous, std::regex_constants::match_not_null, std::regex_constants::match_prev_avail, and std::regex_search().

operator++() [2/2]

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
regex_iterator std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++ (
            int )  [inline]
```

Postincrements a regex_iterator.

operator->()

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
const value_type * std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator-> () const
[inline], [noexcept]
```

Selects a regex_iterator member.

operator=()

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
regex_iterator & std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator= (
            const regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > & )  [default]
```

Copy assigns one regex_iterator to another.

operator==()

```
template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>
bool std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator== (
            const regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs) const  [noexcept]
```

Tests the equivalence of two regex iterators.

References regex_iterator().

The documentation for this class was generated from the following files:

- regex.h
- regex.tcc

5.847 std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > Class Template Reference

```
#include <regex>
```

Public Types

- typedef std::ptrdiff_t **difference_type**
- typedef std::forward_iterator_tag **iterator_category**
- typedef std::input_iterator_tag **iterator_concept**

- typedef const value_type * **pointer**
- typedef const value_type & **reference**
- typedef basic_regex< _Ch_type, _Rx_traits > **regex_type**
- typedef sub_match< _Bi_iter > **value_type**

Public Member Functions

- regex_token_iterator ()
- template<std::size_t _Nm>
  regex_token_iterator (_Bi_iter __a, _Bi_iter __b, const regex_type &__re, const int(&__submatches)[_Nm], regex_constants::match_flag_type __m=regex_constants::match_default)
- regex_token_iterator (_Bi_iter __a, _Bi_iter __b, const regex_type &__re, const std::vector< int > &__↩ submatches, regex_constants::match_flag_type __m=regex_constants::match_default)
- regex_token_iterator (_Bi_iter __a, _Bi_iter __b, const regex_type &__re, initializer_list< int > __submatches, regex_constants::match_flag_type __m=regex_constants::match_default)
- regex_token_iterator (_Bi_iter __a, _Bi_iter __b, const regex_type &__re, int __submatch=0, regex_constants::match_flag_type __m=regex_constants::match_default)
- template<std::size_t _Nm>
  regex_token_iterator (_Bi_iter, _Bi_iter, const regex_type &&, const int(&)[_Nm], regex_constants::match_flag_type=regex_consta
- **regex_token_iterator** (_Bi_iter, _Bi_iter, const regex_type &&, const std::vector< int > &, regex_constants::match_flag_type=regex
- **regex_token_iterator** (_Bi_iter, _Bi_iter, const regex_type &&, initializer_list< int >, regex_constants::match_flag_type=regex_cons
- **regex_token_iterator** (_Bi_iter, _Bi_iter, const regex_type &&, int=0, regex_constants::match_flag_type=regex_constants::match_c
- regex_token_iterator (const regex_token_iterator &__rhs)
- const value_type & operator* () const
- regex_token_iterator & operator++ ()
- regex_token_iterator operator++ (int)
- const value_type * operator-> () const
- regex_token_iterator & operator= (const regex_token_iterator &__rhs)
- bool operator== (const regex_token_iterator &__rhs) const
- bool **operator==** (default_sentinel_t) const noexcept

5.847.1 Detailed Description

template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>>
class std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >

Iterates over submatches in a range (or *splits* a text string).
The purpose of this iterator is to enumerate all, or all specified, matches of a regular expression within a text range. The dereferenced value of an iterator of this class is a std::sub_match object.

Since

C++11

5.847.2 Constructor & Destructor Documentation

regex_token_iterator() [1/6]

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator ()  [inline]
```

Default constructs a regex_token_iterator.
A default-constructed regex_token_iterator is a singular iterator that will compare equal to the one-past-the-end value for any iterator of the same type.
Referenced by operator++(), operator=(), and operator==().

regex_token_iterator() [2/6]

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator (
            _Bi_iter __a,
            _Bi_iter __b,
            const regex_type & __re,
            int __submatch = 0,
            regex_constants::match_flag_type __m = regex_constants::match_default)  [inline]
```

Constructs a regex_token_iterator...

Parameters

__a	[IN] The start of the text to search.
__b	[IN] One-past-the-end of the text to search.
__re	[IN] The regular expression to search for.
__submatch	[IN] Which submatch to return. There are some special values for this parameter: • -1 each enumerated subexpression does NOT match the regular expression (aka field splitting) • 0 the entire string matching the subexpression is returned for each match within the text. • >0 enumerates only the indicated subexpression from a match within the text.
__m	[IN] Policy flags for match rules.

regex_token_iterator() [3/6]

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator (
            _Bi_iter __a,
            _Bi_iter __b,
            const regex_type & __re,
            const std::vector< int > & __submatches,
            regex_constants::match_flag_type __m = regex_constants::match_default)  [inline]
```

Constructs a regex_token_iterator...

Parameters

__a	[IN] The start of the text to search.
__b	[IN] One-past-the-end of the text to search.
__re	[IN] The regular expression to search for.
__submatches	[IN] A list of subexpressions to return for each regular expression match within the text.
__m	[IN] Policy flags for match rules.

regex_token_iterator() [4/6]

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator (
```

```
            _Bi_iter __a,
            _Bi_iter __b,
            const regex_type & __re,
            initializer_list< int > __submatches,
            regex_constants::match_flag_type __m = regex_constants::match_default)  [inline]
```

Constructs a regex_token_iterator...

Parameters

__a	[IN] The start of the text to search.
__b	[IN] One-past-the-end of the text to search.
__re	[IN] The regular expression to search for.
__submatches	[IN] A list of subexpressions to return for each regular expression match within the text.
__m	[IN] Policy flags for match rules.

regex_token_iterator() [5/6]

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
template<std::size_t _Nm>
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator (
            _Bi_iter __a,
            _Bi_iter __b,
            const regex_type & __re,
            const int(&) __submatches[_Nm],
            regex_constants::match_flag_type __m = regex_constants::match_default)  [inline]
```

Constructs a regex_token_iterator...

Parameters

__a	[IN] The start of the text to search.
__b	[IN] One-past-the-end of the text to search.
__re	[IN] The regular expression to search for.
__submatches	[IN] A list of subexpressions to return for each regular expression match within the text.
__m	[IN] Policy flags for match rules.

regex_token_iterator() [6/6]

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator (
            const regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs)  [inline]
```

Copy constructs a regex_token_iterator.

Parameters

__rhs	[IN] A regex_token_iterator to copy.

5.847.3 Member Function Documentation

operator*()

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
const value_type & std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator* ()
const  [inline]
```

Dereferences a regex_token_iterator.

operator++() [1/2]

```
template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>
regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > & std::regex_token_iterator< _Bi_iter, _↩
Ch_type, _Rx_traits >::operator++ ()
```

Increments a regex_token_iterator.
References regex_token_iterator().

operator++() [2/2]

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
regex_token_iterator std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++ (
            int )  [inline]
```

Postincrements a regex_token_iterator.

operator->()

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
const value_type * std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator-> ()
const  [inline]
```

Selects a regex_token_iterator member.

operator=()

```
template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>
regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > & std::regex_token_iterator< _Bi_iter, _↩
Ch_type, _Rx_traits >::operator= (
            const regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs)
```

Assigns a regex_token_iterator to another.

Parameters

__rhs	[IN] A regex_token_iterator to copy.

References regex_token_iterator().

operator==()

```
template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>
bool std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator== (
            const regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs) const
```

Compares a regex_token_iterator to another for equality.
References regex_token_iterator().
The documentation for this class was generated from the following files:

- regex.h
- regex.tcc

5.848 std::regex_traits< _Ch_type > Class Template Reference

`#include <regex>`

Public Types

- typedef _RegexMask **char_class_type**
- typedef _Ch_type **char_type**
- typedef std::locale **locale_type**
- typedef std::basic_string< char_type > **string_type**

Public Member Functions

- regex_traits ()
- locale_type getloc () const
- locale_type imbue (locale_type __loc)
- bool isctype (_Ch_type __c, char_class_type __f) const
- template<typename _Fwd_iter>
  char_class_type lookup_classname (_Fwd_iter __first, _Fwd_iter __last, bool __icase=false) const
- template<typename _Fwd_iter>
  string_type lookup_collatename (_Fwd_iter __first, _Fwd_iter __last) const
- template<typename _Fwd_iter>
  string_type transform (_Fwd_iter __first, _Fwd_iter __last) const
- template<typename _Fwd_iter>
  string_type transform_primary (_Fwd_iter __first, _Fwd_iter __last) const
- char_type translate (char_type __c) const
- char_type translate_nocase (char_type __c) const
- int value (_Ch_type __ch, int __radix) const

Static Public Member Functions

- static std::size_t length (const char_type *__p)

Protected Attributes

- locale_type **_M_locale**

5.848.1 Detailed Description

template<typename _Ch_type>
class std::regex_traits< _Ch_type >

Describes aspects of a regular expression.
A regular expression traits class that satisfies the requirements of section [28.7].
The class regex is parameterized around a set of related types and functions used to complete the definition of its semantics. This class satisfies the requirements of such a traits class.

Since

C++11

5.848.2 Constructor & Destructor Documentation

regex_traits()

```
template<typename _Ch_type>
std::regex_traits< _Ch_type >::regex_traits ()  [inline]
```

Constructs a default traits object.

5.848.3 Member Function Documentation

getloc()

```
template<typename _Ch_type>
locale_type std::regex_traits< _Ch_type >::getloc () const  [inline]
```

Gets a copy of the current locale in use by the regex_traits object.
Referenced by std::regex_traits< _CharT >::lookup_classname().

imbue()

```
template<typename _Ch_type>
locale_type std::regex_traits< _Ch_type >::imbue (
            locale_type __loc)  [inline]
```

Imbues the regex_traits object with a copy of a new locale.

Parameters

__loc	A locale.

Returns

a copy of the previous locale in use by the regex_traits object.

Note

Calling imbue with a different locale than the one currently in use invalidates all cached data held by *this.

isctype()

```
template<typename _Ch_type>
bool std::regex_traits< _Ch_type >::isctype (
            _Ch_type __c,
            char_class_type __f) const
```

Determines if c is a member of an identified class.

Parameters

_c	a character.
_f	a class type (as returned from lookup_classname).

Returns

true if the character __c is a member of the classification represented by __f, false otherwise.

Exceptions

std::bad_cast	if the current locale does not have a ctype facet.

References std::use_facet().

length()

```
template<typename _Ch_type>
static std::size_t std::regex_traits< _Ch_type >::length (
            const char_type * __p)  [inline], [static]
```

Gives the length of a C-style string starting at __p.

Parameters

_↩ _p	a pointer to the start of a character sequence.

Returns

the number of characters between *__p and the first default-initialized value of type char_type. In other words, uses the C-string algorithm for determining the length of a sequence of characters.

lookup_classname()

```
template<typename _Ch_type>
template<typename _Fwd_iter>
regex_traits< _Ch_type >::char_class_type std::regex_traits< _Ch_type >::lookup_classname (
            _Fwd_iter __first,
            _Fwd_iter __last,
            bool __icase = false) const
```

Maps one or more characters to a named character classification.

Parameters

__first	beginning of the character sequence.
__last	one-past-the-end of the character sequence.
__icase	ignores the case of the classification name.

Returns

an unspecified value that represents the character classification named by the character sequence designated by the iterator range [__first, __last). If icase is true, the returned mask identifies the classification regardless of the case of the characters to be matched (for example, [[:lower:]] is the same as [[:alpha:]]), otherwise a case-dependent classification is returned. The value returned shall be independent of the case of the characters in the character sequence. If the name is not recognized then returns a value that compares equal to 0.

At least the following names (or their wide-character equivalent) are supported.

- d
- w
- s

- alnum
- alpha
- blank
- cntrl
- digit
- graph
- lower
- print
- punct
- space
- upper
- xdigit

References std::use_facet().

lookup_collatename()

```
template<typename _Ch_type>
template<typename _Fwd_iter>
regex_traits< _Ch_type >::string_type std::regex_traits< _Ch_type >::lookup_collatename (
            _Fwd_iter __first,
            _Fwd_iter __last) const
```

Gets a collation element by name.

Parameters

__first	beginning of the collation element name.
__last	one-past-the-end of the collation element name.

Returns

a sequence of one or more characters that represents the collating element consisting of the character sequence designated by the iterator range [__first, __last). Returns an empty string if the character sequence is not a valid collating element.

References std::use_facet().

transform()

```
template<typename _Ch_type>
template<typename _Fwd_iter>
string_type std::regex_traits< _Ch_type >::transform (
            _Fwd_iter __first,
            _Fwd_iter __last) const  [inline]
```

Gets a sort key for a character sequence.

Parameters

__first	beginning of the character sequence.
__last	one-past-the-end of the character sequence.

Returns a sort key for the character sequence designated by the iterator range [F1, F2) such that if the character sequence [G1, G2) sorts before the character sequence [H1, H2) then v.transform(G1, G2) $<$ v.transform(H1, H2). What this really does is provide a more efficient way to compare a string to multiple other strings in locales with fancy collation rules and equivalence classes.

Returns

a locale-specific sort key equivalent to the input range.

Exceptions

std::bad_cast	if the current locale does not have a collate facet.

transform_primary()

```
template<typename _Ch_type>
template<typename _Fwd_iter>
string_type std::regex_traits< _Ch_type >::transform_primary (
            _Fwd_iter __first,
            _Fwd_iter __last) const  [inline]
```

Gets a sort key for a character sequence, independent of case.

Parameters

__first	beginning of the character sequence.
__last	one-past-the-end of the character sequence.

Effects: if `typeid(use_facet<collate<_Ch_type>>(getloc())) == typeid(collate_byname<_Ch_type>)` and the form of the sort key returned by `collate_byname<_Ch_type>::transform(__first, __last)` is known and can be converted into a primary sort key then returns that key, otherwise returns an empty string.

Todo Implement this function correctly.

translate()

```
template<typename _Ch_type>
char_type std::regex_traits< _Ch_type >::translate (
            char_type __c) const  [inline]
```

Performs the identity translation.

Parameters

__c	A character to the locale-specific character set.

Returns

__c.

translate_nocase()

```
template<typename _Ch_type>
char_type std::regex_traits< _Ch_type >::translate_nocase (
            char_type __c) const  [inline]
```

Translates a character into a case-insensitive equivalent.

Parameters

_↩ _c	A character to the locale-specific character set.

Returns

the locale-specific lower-case equivalent of __c.

Exceptions

std::bad_cast	if the imbued locale does not support the ctype facet.

value()

```
template<typename _Ch_type>
int std::regex_traits< _Ch_type >::value (
            _Ch_type __ch,
            int __radix) const
```

Converts a digit to an int.

Parameters

__ch	a character representing a digit.
__radix	the radix if the numeric conversion (limited to 8, 10, or 16).

Returns

the value represented by the digit __ch in base radix if the character __ch is a valid digit in base radix; otherwise returns -1.

References std::basic_ios< _CharT, _Traits >::fail(), std::hex(), and std::oct().
Referenced by std::regex_traits< _CharT >::isctype().
The documentation for this class was generated from the following files:

- regex.h
- regex.tcc

5.849 std::remove_all_extents< _Tp > Struct Template Reference

```
#include <type_traits>
```

Public Types

- using **type**

5.849.1 Detailed Description

template<**typename _Tp**>
struct std::remove_all_extents< **_Tp** >

remove_all_extents
The documentation for this struct was generated from the following file:

- type_traits

5.850 std::remove_const< _Tp > Struct Template Reference

`#include <type_traits>`

Public Types

- using **type**

5.850.1 Detailed Description

template<**typename _Tp**>
struct std::remove_const< **_Tp** >

remove_const
The documentation for this struct was generated from the following file:

- type_traits

5.851 std::remove_cv< _Tp > Struct Template Reference

`#include <type_traits>`

Public Types

- using **type**

5.851.1 Detailed Description

template<**typename _Tp**>
struct std::remove_cv< **_Tp** >

remove_cv
The documentation for this struct was generated from the following file:

- type_traits

5.852 std::remove_extent< _Tp > Struct Template Reference

`#include <type_traits>`

Public Types

- using **type**

5.852.1 Detailed Description

template<typename _Tp>
struct std::remove_extent< _Tp >

remove_extent
The documentation for this struct was generated from the following file:

- type_traits

5.853 std::remove_pointer< _Tp > Struct Template Reference

`#include <type_traits>`

Public Types

- using **type**

5.853.1 Detailed Description

template<typename _Tp>
struct std::remove_pointer< _Tp >

remove_pointer
The documentation for this struct was generated from the following file:

- type_traits

5.854 std::remove_reference< _Tp > Struct Template Reference

`#include <type_traits>`

Public Types

- using **type**

5.854.1 Detailed Description

template<typename _Tp>
struct std::remove_reference< _Tp >

remove_reference
The documentation for this struct was generated from the following file:

- type_traits

5.855 std::remove_volatile< _Tp > Struct Template Reference

`#include <type_traits>`

Public Types

- using **type**

5.855.1 Detailed Description

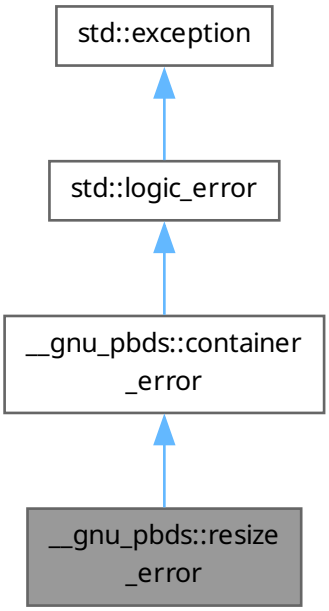
template<typename _Tp>
struct std::remove_volatile< _Tp >

remove_volatile
The documentation for this struct was generated from the following file:

- type_traits

5.856 __gnu_pbds::resize_error Struct Reference

`#include <exception.hpp>`
Inheritance diagram for __gnu_pbds::resize_error:



Public Member Functions

- virtual const char * what () const noexcept

5.856.1 Detailed Description

A container cannot be resized.

5.856.2 Member Function Documentation

what()

`virtual const char * std::logic_error::what () const [virtual], [noexcept], [inherited]`
Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).

Reimplemented from std::exception.
Reimplemented in std::future_error.
The documentation for this struct was generated from the following file:

- exception.hpp

5.857 __gnu_pbds::detail::resize_policy< _Tp > Class Template Reference

`#include <resize_policy.hpp>`

Public Types

- typedef _Tp **size_type**

Public Member Functions

- **resize_policy** (const resize_policy &other)
- size_type **get_new_size_for_arbitrary** (size_type) const
- size_type **get_new_size_for_grow** () const
- size_type **get_new_size_for_shrink** () const
- bool **grow_needed** (size_type) const
- void **notify_arbitrary** (size_type)
- void **notify_grow_resize** ()
- void **notify_shrink_resize** ()
- bool **resize_needed_for_grow** (size_type) const
- bool **resize_needed_for_shrink** (size_type) const
- bool **shrink_needed** (size_type) const
- void **swap** (resize_policy< _Tp > &)

Static Public Attributes

- static const _Tp **min_size**

5.857.1 Detailed Description

template<typename _Tp>
class __gnu_pbds::detail::resize_policy< _Tp >

Resize policy for binary heap.
The documentation for this class was generated from the following file:

- resize_policy.hpp

5.858 std::result_of< _Signature > Struct Template Reference

5.858.1 Detailed Description

template<typename _Signature>
struct std::result_of< _Signature >

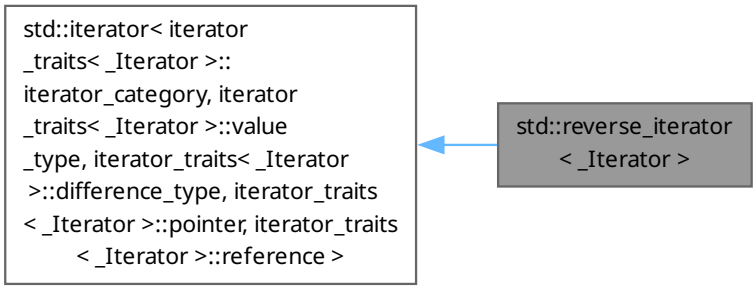
result_of
The documentation for this struct was generated from the following file:

- type_traits

5.859 std::reverse_iterator< _Iterator > Class Template Reference

`#include <stl_iterator.h>`

Inheritance diagram for std::reverse_iterator< _Iterator >:



Public Types

- using **difference_type**
- using **iterator_category**
- using **iterator_concept**
- typedef _Iterator **iterator_type**
- typedef __traits_type::pointer **pointer**
- using **reference**
- using **value_type**

Public Member Functions

- constexpr reverse_iterator () noexcept(/*conditional */)
- constexpr reverse_iterator (const reverse_iterator &__x) noexcept(/*conditional */)
- template<typename _Iter>
  requires __convertible<_Iter>
  constexpr reverse_iterator (const reverse_iterator< _Iter > &__x) noexcept(/*conditional */)
- constexpr reverse_iterator (iterator_type __x) noexcept(/*conditional */)
- constexpr iterator_type base () const noexcept(/*conditional */)
- constexpr reference operator* () const
- constexpr reverse_iterator operator+ (difference_type __n) const
- constexpr reverse_iterator & operator++ ()
- constexpr reverse_iterator operator++ (int)
- constexpr reverse_iterator & operator+= (difference_type __n)
- constexpr reverse_iterator operator- (difference_type __n) const
- constexpr reverse_iterator & operator-- ()
- constexpr reverse_iterator operator-- (int)
- constexpr reverse_iterator & operator-= (difference_type __n)
- constexpr pointer operator-> () const
- reverse_iterator & **operator=** (const reverse_iterator &)=default
- template<typename _Iter>
  requires __convertible<_Iter> && assignable_from<_Iterator&, const _Iter&>
  constexpr reverse_iterator & **operator=** (const reverse_iterator< _Iter > &__x) noexcept(/*conditional */)
- constexpr reference operator[ ] (difference_type __n) const

Protected Types

- typedef iterator_traits< _Iterator > **__traits_type**

Protected Attributes

- _Iterator **current**

Friends

- constexpr iter_rvalue_reference_t< _Iterator > **iter_move** (const reverse_iterator &__i) noexcept(is_nothrow_↩ copy_constructible_v< _Iterator > &&noexcept(ranges::iter_move(--std::declval< _Iterator & >())))
- template<indirectly_swappable< _Iterator > _Iter2>
  constexpr void **iter_swap** (const reverse_iterator &__x, const reverse_iterator< _Iter2 > &__y) noexcept(is_↩ nothrow_copy_constructible_v< _Iterator > &&is_nothrow_copy_constructible_v< _Iter2 > &&noexcept(ranges↩ ::iter_swap(--std::declval< _Iterator & >(), --std::declval< _Iter2 & >())))

5.859.1 Detailed Description

template<typename _Iterator>
class std::reverse_iterator< _Iterator >

Bidirectional and random access iterators have corresponding reverse iterator adaptors that iterate through the data structure in the opposite direction. They have the same signatures as the corresponding iterators. The fundamental relation between a reverse iterator and its corresponding iterator `i` is established by the identity:

```
&*(reverse_iterator(i)) == &*(i - 1)
```

This mapping is dictated by the fact that while there is always a pointer past the end of an array, there might not be a valid pointer before the beginning of an array. [24.4.1]/1,2

Reverse iterators can be tricky and surprising at first. Their semantics make sense, however, and the trickiness is a side effect of the requirement that the iterators must be safe.

5.859.2 Constructor & Destructor Documentation

reverse_iterator() [1/4]

```
template<typename _Iterator>
std::reverse_iterator< _Iterator >::reverse_iterator ()  [inline], [constexpr], [noexcept]
```

The default constructor value-initializes member `current`. If it is a pointer, that means it is zero-initialized.

reverse_iterator() [2/4]

```
template<typename _Iterator>
std::reverse_iterator< _Iterator >::reverse_iterator (
            iterator_type __x)  [inline], [explicit], [constexpr], [noexcept]
```

This iterator will move in the opposite direction that `x` does.

reverse_iterator() [3/4]

```
template<typename _Iterator>
std::reverse_iterator< _Iterator >::reverse_iterator (
            const reverse_iterator< _Iterator > & __x)  [inline], [constexpr], [noexcept]
```

The copy constructor is normal.

reverse_iterator() [4/4]

```
template<typename _Iterator>
template<typename _Iter>
requires __convertible<_Iter>
std::reverse_iterator< _Iterator >::reverse_iterator (
            const reverse_iterator< _Iter > & __x)  [inline], [constexpr], [noexcept]
```

A reverse_iterator across other types can be copied if the underlying iterator can be converted to the type of `current`.

5.859.3 Member Function Documentation

base()

```
template<typename _Iterator>
iterator_type std::reverse_iterator< _Iterator >::base () const  [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns

`current`, the iterator used for underlying work.

Referenced by std::operator==().

operator*()

```
template<typename _Iterator>
reference std::reverse_iterator< _Iterator >::operator* () const  [inline], [nodiscard], [constexpr]
```

Returns

A reference to the value at `--current`

This requires that `--current` is dereferenceable.

Warning

This implementation requires that for an iterator of the underlying iterator type, `x`, a reference obtained by `*x` remains valid after `x` has been modified or destroyed. This is a bug: http://gcc.gnu.org/PR51823

operator+()

```
template<typename _Iterator>
reverse_iterator std::reverse_iterator< _Iterator >::operator+ (
            difference_type __n) const  [inline], [nodiscard], [constexpr]
```

Returns

A reverse_iterator that refers to `current` - *__n*

The underlying iterator must be a Random Access Iterator.

operator++() [1/2]

```
template<typename _Iterator>
reverse_iterator & std::reverse_iterator< _Iterator >::operator++ ()  [inline], [constexpr]
```

Returns

`*this`

Decrements the underlying iterator.

operator++() [2/2]

```
template<typename _Iterator>
reverse_iterator std::reverse_iterator< _Iterator >::operator++ (
            int )  [inline], [constexpr]
```

Returns

The original value of *this

Decrements the underlying iterator.

operator+=()

```
template<typename _Iterator>
reverse_iterator & std::reverse_iterator< _Iterator >::operator+= (
            difference_type __n)  [inline], [constexpr]
```

Returns

*this

Moves the underlying iterator backwards *__n* steps. The underlying iterator must be a Random Access Iterator.

operator-()

```
template<typename _Iterator>
reverse_iterator std::reverse_iterator< _Iterator >::operator- (
            difference_type __n) const  [inline], [nodiscard], [constexpr]
```

Returns

A reverse_iterator that refers to `current` - *__n*

The underlying iterator must be a Random Access Iterator.

operator--() [1/2]

```
template<typename _Iterator>
reverse_iterator & std::reverse_iterator< _Iterator >::operator-- ()  [inline], [constexpr]
```

Returns

*this

Increments the underlying iterator.

operator--() [2/2]

```
template<typename _Iterator>
reverse_iterator std::reverse_iterator< _Iterator >::operator-- (
            int )  [inline], [constexpr]
```

Returns

A reverse_iterator with the previous value of *this

Increments the underlying iterator.

operator-=()

```
template<typename _Iterator>
reverse_iterator & std::reverse_iterator< _Iterator >::operator-= (
            difference_type __n)  [inline], [constexpr]
```

Returns

*this

Moves the underlying iterator forwards __*n* steps. The underlying iterator must be a Random Access Iterator.

operator->()

```
template<typename _Iterator>
pointer std::reverse_iterator< _Iterator >::operator-> () const  [inline], [nodiscard], [constexpr]
```

Returns

A pointer to the value at `--current`

This requires that `--current` is dereferenceable.

operator[]()

```
template<typename _Iterator>
reference std::reverse_iterator< _Iterator >::operator[] (
            difference_type __n) const  [inline], [nodiscard], [constexpr]
```

Returns

The value at `current` - __*n* - 1

The underlying iterator must be a Random Access Iterator.
The documentation for this class was generated from the following file:

- bits/stl_iterator.h

5.860 __gnu_cxx::rope< _CharT, _Alloc > Class Template Reference

`#include <rope>`

Public Types

- typedef _Rope_RopeConcatenation< _CharT, _Alloc > **__C**
- typedef _Rope_RopeFunction< _CharT, _Alloc > **__F**
- typedef _Rope_RopeLeaf< _CharT, _Alloc > **__L**
- typedef _Rope_RopeSubstring< _CharT, _Alloc > **__S**
- typedef __alloc_traits< _Alloc >::template rebind< __C >::other **_CAlloc**
- typedef __alloc_traits< _Alloc >::template rebind< _CharT >::other **_DataAlloc**
- typedef __alloc_traits< _Alloc >::template rebind< __F >::other **_FAlloc**
- typedef __alloc_traits< _Alloc >::template rebind< __L >::other **_LAlloc**
- typedef __alloc_traits< _Alloc >::template rebind< __S >::other **_SAlloc**
- typedef _Rope_const_iterator< _CharT, _Alloc > **const_iterator**
- typedef const _CharT * **const_pointer**
- typedef _CharT **const_reference**
- typedef std::reverse_iterator< const_iterator > **const_reverse_iterator**
- typedef std::ptrdiff_t **difference_type**

- typedef _Rope_iterator< _CharT, _Alloc > **iterator**
- typedef _Rope_char_ptr_proxy< _CharT, _Alloc > **pointer**
- typedef _Rope_char_ref_proxy< _CharT, _Alloc > **reference**
- typedef std::reverse_iterator< iterator > **reverse_iterator**
- typedef std::size_t **size_type**
- typedef _CharT **value_type**

Public Member Functions

- **rope** (_CharT __c, const allocator_type &__a=allocator_type())
- **rope** (char_producer< _CharT > *__fn, size_type __len, bool __delete_fn, const allocator_type &__a=allocator_type())
- **rope** (const _CharT *__s, const _CharT *__e, const allocator_type &__a=allocator_type())
- **rope** (const _CharT *__s, const allocator_type &__a=allocator_type())
- **rope** (const _CharT *__s, size_type __len, const allocator_type &__a=allocator_type())
- **rope** (const allocator_type &__a=allocator_type())
- **rope** (const const_iterator &__s, const const_iterator &__e, const allocator_type &__a=allocator_type())
- **rope** (const iterator &__s, const iterator &__e, const allocator_type &__a=allocator_type())
- **rope** (const rope &__x, const allocator_type &__a=allocator_type())
- **rope** (size_type __n, _CharT __c, const allocator_type &__a=allocator_type())
- allocator_type & **_M_get_allocator** ()
- const allocator_type & **_M_get_allocator** () const
- rope & **append** ()
- rope & **append** (_CharT __c)
- rope & **append** (const _CharT *__c_string)
- rope & **append** (const _CharT *__iter, size_type __n)
- rope & **append** (const _CharT *__s, const _CharT *__e)
- rope & **append** (const rope &__y)
- rope & **append** (const_iterator __s, const_iterator __e)
- rope & **append** (size_type __n, _CharT __c)
- void **apply_to_pieces** (size_type __begin, size_type __end, _Rope_char_consumer< _CharT > &__c) const
- _CharT **at** (size_type __pos) const
- _CharT **back** () const
- void **balance** ()
- const_iterator **begin** ()
- const_iterator **begin** () const
- const _CharT * **c_str** () const
- void **clear** ()
- int **compare** (const rope &__y) const
- const_iterator **const_begin** () const
- const_iterator **const_end** () const
- const_reverse_iterator **const_rbegin** () const
- const_reverse_iterator **const_rend** () const
- void **copy** (_CharT *__buffer) const
- size_type **copy** (size_type __pos, size_type __n, _CharT *__buffer) const
- void **delete_c_str** ()
- void **dump** ()
- bool **empty** () const
- const_iterator **end** ()
- const_iterator **end** () const
- iterator **erase** (const iterator &__p)

- iterator **erase** (const iterator &__p, const iterator &__q)
- void **erase** (size_type __p, size_type __n)
- size_type **find** (_CharT __c, size_type __pos=0) const
- size_type **find** (const _CharT *__s, size_type __pos=0) const
- _CharT **front** () const
- allocator_type **get_allocator** () const
- iterator **insert** (const iterator &__p)
- iterator **insert** (const iterator &__p, _CharT __c)
- iterator **insert** (const iterator &__p, const _CharT *__i, const _CharT *__j)
- iterator **insert** (const iterator &__p, const _CharT *__i, size_type __n)
- iterator **insert** (const iterator &__p, const _CharT *c_string)
- iterator **insert** (const iterator &__p, const const_iterator &__i, const const_iterator &__j)
- iterator **insert** (const iterator &__p, const iterator &__i, const iterator &__j)
- iterator **insert** (const iterator &__p, const rope &__r)
- iterator **insert** (const iterator &__p, size_type __n, _CharT __c)
- void **insert** (size_type __p)
- void **insert** (size_type __p, _CharT __c)
- void **insert** (size_type __p, const _CharT *__c_string)
- void **insert** (size_type __p, const _CharT *__i, const _CharT *__j)
- void **insert** (size_type __p, const _CharT *__i, size_type __n)
- void **insert** (size_type __p, const const_iterator &__i, const const_iterator &__j)
- void **insert** (size_type __p, const iterator &__i, const iterator &__j)
- void **insert** (size_type __p, const rope &__r)
- void **insert** (size_type __p, size_type __n, _CharT __c)
- size_type **length** () const
- size_type **max_size** () const
- iterator **mutable_begin** ()
- iterator **mutable_end** ()
- reverse_iterator **mutable_rbegin** ()
- reference **mutable_reference_at** (size_type __pos)
- reverse_iterator **mutable_rend** ()
- rope & **operator=** (const rope &__x)
- _CharT **operator[ ]** (size_type __pos) const
- void **pop_back** ()
- void **pop_front** ()
- void **push_back** (_CharT __x)
- void **push_front** (_CharT __x)
- const_reverse_iterator **rbegin** ()
- const_reverse_iterator **rbegin** () const
- const_reverse_iterator **rend** ()
- const_reverse_iterator **rend** () const
- void **replace** (const iterator &__p, _CharT __c)
- void **replace** (const iterator &__p, const _CharT *__c_string)
- void **replace** (const iterator &__p, const _CharT *__i, const _CharT *__j)
- void **replace** (const iterator &__p, const _CharT *__i, size_type __n)
- void **replace** (const iterator &__p, const iterator &__q, _CharT __c)
- void **replace** (const iterator &__p, const iterator &__q, const _CharT *__c_string)
- void **replace** (const iterator &__p, const iterator &__q, const _CharT *__i, const _CharT *__j)
- void **replace** (const iterator &__p, const iterator &__q, const _CharT *__i, size_type __n)
- void **replace** (const iterator &__p, const iterator &__q, const const_iterator &__i, const const_iterator &__j)
- void **replace** (const iterator &__p, const iterator &__q, const iterator &__i, const iterator &__j)

- void **replace** (const iterator &__p, const iterator &__q, const rope &__r)
- void **replace** (const iterator &__p, const rope &__r)
- void **replace** (const iterator &__p, const_iterator __i, const_iterator __j)
- void **replace** (const iterator &__p, iterator __i, iterator __j)
- void **replace** (size_type __p, _CharT __c)
- void **replace** (size_type __p, const _CharT *__c_string)
- void **replace** (size_type __p, const _CharT *__i, const _CharT *__j)
- void **replace** (size_type __p, const _CharT *__i, size_type __i_len)
- void **replace** (size_type __p, const const_iterator &__i, const const_iterator &__j)
- void **replace** (size_type __p, const iterator &__i, const iterator &__j)
- void **replace** (size_type __p, const rope &__r)
- void **replace** (size_type __p, size_type __n, _CharT __c)
- void **replace** (size_type __p, size_type __n, const _CharT *__c_string)
- void **replace** (size_type __p, size_type __n, const _CharT *__i, const _CharT *__j)
- void **replace** (size_type __p, size_type __n, const _CharT *__i, size_type __i_len)
- void **replace** (size_type __p, size_type __n, const const_iterator &__i, const const_iterator &__j)
- void **replace** (size_type __p, size_type __n, const iterator &__i, const iterator &__j)
- void **replace** (size_type __p, size_type __n, const rope &__r)
- const _CharT * **replace_with_c_str** ()
- size_type **size** () const
- rope< _CharT, _Alloc > **substr** (const_iterator __start)
- rope **substr** (const_iterator __start, const_iterator __end) const
- rope **substr** (iterator __start) const
- rope **substr** (iterator __start, iterator __end) const
- rope **substr** (size_type __start, size_type __len=1) const
- void **swap** (rope &__b)

Static Public Member Functions

- static __C * **_C_allocate** (std::size_t __n)
- static void **_C_deallocate** (__C *__p, std::size_t __n)
- static _CharT * **_Data_allocate** (std::size_t __n)
- static void **_Data_deallocate** (_CharT *__p, std::size_t __n)
- static __F * **_F_allocate** (std::size_t __n)
- static void **_F_deallocate** (__F *__p, std::size_t __n)
- static __L * **_L_allocate** (std::size_t __n)
- static void **_L_deallocate** (__L *__p, std::size_t __n)
- static __S * **_S_allocate** (std::size_t __n)
- static void **_S_deallocate** (__S *__p, std::size_t __n)

Public Attributes

- _RopeRep * **_M_tree_ptr**

Static Public Attributes

- static const size_type **npos**

Protected Types

- enum { **_S_copy_max** }
- typedef _Rope_base< _CharT, _Alloc > **_Base**
- typedef _CharT * **_Cstrptr**
- typedef _Rope_RopeConcatenation< _CharT, _Alloc > **_RopeConcatenation**
- typedef _Rope_RopeFunction< _CharT, _Alloc > **_RopeFunction**
- typedef _Rope_RopeLeaf< _CharT, _Alloc > **_RopeLeaf**
- typedef _Rope_RopeRep< _CharT, _Alloc > **_RopeRep**
- typedef _Rope_RopeSubstring< _CharT, _Alloc > **_RopeSubstring**
- typedef _Rope_self_destruct_ptr< _CharT, _Alloc > **_Self_destruct_ptr**
- typedef _Base::allocator_type **allocator_type**

Static Protected Member Functions

- static size_type **_S_allocated_capacity** (size_type __n)
- static bool **_S_apply_to_pieces** (_Rope_char_consumer< _CharT > &__c, const _RopeRep *__r, size_type __begin, size_type __end)
- static _RopeRep * **_S_concat** (_RopeRep *__left, _RopeRep *__right)
- static _RopeRep * **_S_concat_char_iter** (_RopeRep *__r, const _CharT *__iter, size_type __slen, allocator_type &__a)
- static _RopeRep * **_S_destr_concat_char_iter** (_RopeRep *__r, const _CharT *__iter, size_type __slen, allocator_type &__a)
- static _RopeLeaf * **_S_destr_leaf_concat_char_iter** (_RopeLeaf *__r, const _CharT *__iter, size_type __slen)
- static _CharT **_S_fetch** (_RopeRep *__r, size_type __pos)
- static _CharT * **_S_fetch_ptr** (_RopeRep *__r, size_type __pos)
- static bool **_S_is0** (_CharT __c)
- static _RopeLeaf * **_S_leaf_concat_char_iter** (_RopeLeaf *__r, const _CharT *__iter, size_type __slen)
- static _RopeConcatenation * **_S_new_RopeConcatenation** (_RopeRep *__left, _RopeRep *__right, allocator_type &__a)
- static _RopeFunction * **_S_new_RopeFunction** (char_producer< _CharT > *__f, size_type __size, bool __d, allocator_type &__a)
- static _RopeLeaf * **_S_new_RopeLeaf** (_CharT *__s, size_type __size, allocator_type &__a)
- static _RopeSubstring * **_S_new_RopeSubstring** (_Rope_RopeRep< _CharT, _Alloc > *__b, size_type __s, size_type __l, allocator_type &__a)
- static void **_S_ref** (_RopeRep *__t)
- static _RopeLeaf * **_S_RopeLeaf_from_unowned_char_ptr** (const _CharT *__s, size_type __size, allocator_type &__a)
- static size_type **_S_rounded_up_size** (size_type __n)
- static _RopeRep * **_S_substring** (_RopeRep *__base, size_type __start, size_type __endp1)
- static _RopeRep * **_S_tree_concat** (_RopeRep *__left, _RopeRep *__right)
- static void **_S_unref** (_RopeRep *__t)
- static _RopeRep * **replace** (_RopeRep *__old, size_type __pos1, size_type __pos2, _RopeRep *__r)

Static Protected Attributes

- static _CharT **_S_empty_c_str** [1]

Friends

- class **_Rope_char_ptr_proxy**< **_CharT, _Alloc** >
- class **_Rope_char_ref_proxy**< **_CharT, _Alloc** >
- class **_Rope_const_iterator**< **_CharT, _Alloc** >
- class **_Rope_iterator**< **_CharT, _Alloc** >
- class **_Rope_iterator_base**< **_CharT, _Alloc** >
- struct **_Rope_RopeRep**< **_CharT, _Alloc** >
- struct **_Rope_RopeSubstring**< **_CharT, _Alloc** >
- template<class _CharT2, class _Alloc2>
  rope< _CharT2, _Alloc2 > **operator+** (const rope< _CharT2, _Alloc2 > &__left, _CharT2 __right)
- template<class _CharT2, class _Alloc2>
  rope< _CharT2, _Alloc2 > **operator+** (const rope< _CharT2, _Alloc2 > &__left, const _CharT2 *__right)
- template<class _CharT2, class _Alloc2>
  rope< _CharT2, _Alloc2 > **operator+** (const rope< _CharT2, _Alloc2 > &__left, const rope< _CharT2, _Alloc2 > &__right)

5.860.1 Detailed Description

template<class _CharT, class _Alloc>
class __gnu_cxx::rope< _CharT, _Alloc >

This is an SGI extension.

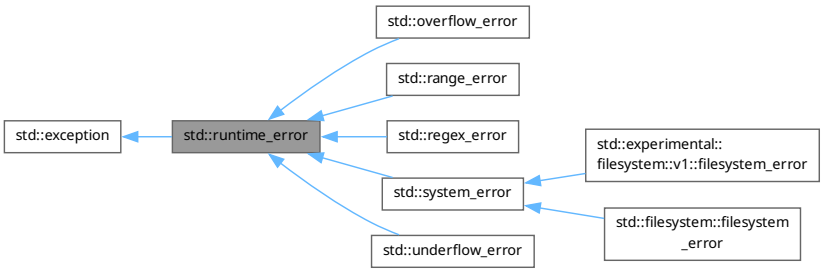
Todo Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

The documentation for this class was generated from the following files:

- rope
- ropeimpl.h

5.861 std::runtime_error Class Reference

`#include <stdexcept>`
Inheritance diagram for std::runtime_error:



Public Member Functions

- **runtime_error** (const char *)
- **runtime_error** (const runtime_error &) noexcept

- runtime_error (const string &__arg)
- **runtime_error** (runtime_error &&) noexcept
- runtime_error & **operator=** (const runtime_error &) noexcept
- runtime_error & **operator=** (runtime_error &&) noexcept
- virtual const char * what () const noexcept

5.861.1 Detailed Description

One of two subclasses of exception.
Runtime errors represent problems outside the scope of a program; they cannot be easily predicted and can generally only be caught as the program executes.

5.861.2 Constructor & Destructor Documentation

runtime_error()

```
std::runtime_error::runtime_error (
            const string & __arg)  [explicit]
```

Takes a character string describing the error.

5.861.3 Member Function Documentation

what()

```
virtual const char * std::runtime_error::what () const  [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).
Reimplemented from std::exception.
Reimplemented in std::experimental::filesystem::v1::filesystem_error, and std::filesystem::filesystem_error.
The documentation for this class was generated from the following file:

- stdexcept

5.862 __gnu_pbds::sample_probe_fn Class Reference

```
#include <sample_probe_fn.hpp>
```

Public Types

- typedef std::size_t **size_type**

Public Member Functions

- sample_probe_fn ()
- sample_probe_fn (const sample_probe_fn &)
- void swap (sample_probe_fn &)

Protected Member Functions

- size_type operator() (key_const_reference r_key, size_type i) const

5.862.1 Detailed Description

A sample probe policy.

5.862.2 Constructor & Destructor Documentation

sample_probe_fn() [1/2]

```
__gnu_pbds::sample_probe_fn::sample_probe_fn ()
```

Default constructor.
Referenced by sample_probe_fn(), and swap().

sample_probe_fn() [2/2]

```
__gnu_pbds::sample_probe_fn::sample_probe_fn (
            const sample_probe_fn & )
```

Copy constructor.
References sample_probe_fn().

5.862.3 Member Function Documentation

operator()()

```
size_type __gnu_pbds::sample_probe_fn::operator() (
            key_const_reference r_key,
            size_type i) const  [inline], [protected]
```

Returns the i-th offset from the hash value of some key r_key.

swap()

```
void __gnu_pbds::sample_probe_fn::swap (
            sample_probe_fn & )  [inline]
```

Swaps content.
References sample_probe_fn().
The documentation for this class was generated from the following file:

- sample_probe_fn.hpp

5.863 __gnu_pbds::sample_range_hashing Class Reference

```
#include <sample_range_hashing.hpp>
```

Public Types

- typedef std::size_t size_type

Public Member Functions

- sample_range_hashing ()
- sample_range_hashing (const sample_range_hashing &other)
- void swap (sample_range_hashing &other)

Protected Member Functions

- void notify_resized (size_type)
- size_type operator() (size_type) const

5.863.1 Detailed Description

A sample range-hashing functor.

5.863.2 Member Typedef Documentation

size_type

```
typedef std::size_t __gnu_pbds::sample_range_hashing::size_type
```
Size type.

5.863.3 Constructor & Destructor Documentation

sample_range_hashing() [1/2]

```
__gnu_pbds::sample_range_hashing::sample_range_hashing ()
```
Default constructor.
Referenced by sample_range_hashing(), and swap().

sample_range_hashing() [2/2]

```
__gnu_pbds::sample_range_hashing::sample_range_hashing (
            const sample_range_hashing & other)
```
Copy constructor.
References sample_range_hashing().

5.863.4 Member Function Documentation

notify_resized()

```
void __gnu_pbds::sample_range_hashing::notify_resized (
            size_type )  [protected]
```
Notifies the policy object that the container's size has changed to argument's size.

operator()()

```
size_type __gnu_pbds::sample_range_hashing::operator() (
            size_type ) const  [inline], [protected]
```
Transforms the __hash value hash into a ranged-hash value.

swap()

```
void __gnu_pbds::sample_range_hashing::swap (
            sample_range_hashing & other)  [inline]
```
Swaps content.
References sample_range_hashing().
The documentation for this class was generated from the following file:

- sample_range_hashing.hpp

5.864 __gnu_pbds::sample_ranged_hash_fn Class Reference

```
#include <sample_ranged_hash_fn.hpp>
```

Public Types

- typedef std::size_t **size_type**

Public Member Functions

- sample_ranged_hash_fn ()
- sample_ranged_hash_fn (const sample_ranged_hash_fn &)
- void swap (sample_ranged_hash_fn &)

Protected Member Functions

- void notify_resized (size_type)
- size_type operator() (key_const_reference) const

5.864.1 Detailed Description

A sample ranged-hash functor.

5.864.2 Constructor & Destructor Documentation

sample_ranged_hash_fn() [1/2]

```
__gnu_pbds::sample_ranged_hash_fn::sample_ranged_hash_fn ()
```

Default constructor.
Referenced by sample_ranged_hash_fn(), and swap().

sample_ranged_hash_fn() [2/2]

```
__gnu_pbds::sample_ranged_hash_fn::sample_ranged_hash_fn (
            const sample_ranged_hash_fn & )
```

Copy constructor.
References sample_ranged_hash_fn().

5.864.3 Member Function Documentation

notify_resized()

```
void __gnu_pbds::sample_ranged_hash_fn::notify_resized (
            size_type )  [protected]
```

Notifies the policy object that the container's __size has changed to size.

operator()()

```
size_type __gnu_pbds::sample_ranged_hash_fn::operator() (
            key_const_reference ) const  [inline], [protected]
```

Transforms key_const_reference into a position within the table.

swap()

```
void __gnu_pbds::sample_ranged_hash_fn::swap (
            sample_ranged_hash_fn & )  [inline]
```

Swaps content.
References sample_ranged_hash_fn().
The documentation for this class was generated from the following file:

- sample_ranged_hash_fn.hpp

5.865 __gnu_pbds::sample_ranged_probe_fn Class Reference

```
#include <sample_ranged_probe_fn.hpp>
```

Public Types

- typedef std::size_t **size_type**

Public Member Functions

- **sample_ranged_probe_fn** (const sample_ranged_probe_fn &)
- void **swap** (sample_ranged_probe_fn &)

Protected Member Functions

- void **notify_resized** (size_type)
- size_type **operator()** (key_const_reference, std::size_t, size_type) const

5.865.1 Detailed Description

A sample ranged-probe functor.
The documentation for this class was generated from the following file:

- sample_ranged_probe_fn.hpp

5.866 __gnu_pbds::sample_resize_policy Class Reference

`#include <sample_resize_policy.hpp>`

Public Types

- typedef std::size_t size_type

Public Member Functions

- sample_resize_policy ()
- sample_range_hashing (const sample_resize_policy &other)
- void swap (sample_resize_policy &other)

Protected Member Functions

- size_type get_new_size (size_type size, size_type num_used_e) const
- bool is_resize_needed () const
- void notify_cleared ()
- void notify_erase_search_collision ()
- void notify_erase_search_end ()
- void notify_erase_search_start ()
- void notify_erased (size_type num_e)
- void notify_find_search_collision ()
- void notify_find_search_end ()
- void notify_find_search_start ()
- void notify_insert_search_collision ()
- void notify_insert_search_end ()
- void notify_insert_search_start ()
- void notify_inserted (size_type num_e)
- void notify_resized (size_type new_size)

5.866.1 Detailed Description

A sample resize policy.

5.866.2 Member Typedef Documentation

size_type

```
typedef std::size_t __gnu_pbds::sample_resize_policy::size_type
```
Size type.

5.866.3 Constructor & Destructor Documentation

sample_resize_policy()

```
__gnu_pbds::sample_resize_policy::sample_resize_policy ()
```
Default constructor.
Referenced by sample_range_hashing(), and swap().

5.866.4 Member Function Documentation

get_new_size()

```
size_type __gnu_pbds::sample_resize_policy::get_new_size (
            size_type size,
            size_type num_used_e) const  [protected]
```
Queries what the new size should be.

is_resize_needed()

```
bool __gnu_pbds::sample_resize_policy::is_resize_needed () const  [inline], [protected]
```
Queries whether a resize is needed.

notify_cleared()

```
void __gnu_pbds::sample_resize_policy::notify_cleared ()  [protected]
```
Notifies the table was cleared.

notify_erase_search_collision()

```
void __gnu_pbds::sample_resize_policy::notify_erase_search_collision ()  [inline], [protected]
```
Notifies a search encountered a collision.

notify_erase_search_end()

```
void __gnu_pbds::sample_resize_policy::notify_erase_search_end ()  [inline], [protected]
```
Notifies a search ended.

notify_erase_search_start()

```
void __gnu_pbds::sample_resize_policy::notify_erase_search_start ()  [inline], [protected]
```
Notifies a search started.

notify_erased()

```
void __gnu_pbds::sample_resize_policy::notify_erased (
            size_type num_e)  [inline], [protected]
```
Notifies an element was erased.

notify_find_search_collision()

```
void __gnu_pbds::sample_resize_policy::notify_find_search_collision ()  [inline], [protected]
```
Notifies a search encountered a collision.

notify_find_search_end()

```
void __gnu_pbds::sample_resize_policy::notify_find_search_end ()  [inline], [protected]
```
Notifies a search ended.

notify_find_search_start()

```
void __gnu_pbds::sample_resize_policy::notify_find_search_start ()  [inline], [protected]
```
Notifies a search started.

notify_insert_search_collision()

```
void __gnu_pbds::sample_resize_policy::notify_insert_search_collision ()  [inline], [protected]
```
Notifies a search encountered a collision.

notify_insert_search_end()

```
void __gnu_pbds::sample_resize_policy::notify_insert_search_end ()  [inline], [protected]
```
Notifies a search ended.

notify_insert_search_start()

```
void __gnu_pbds::sample_resize_policy::notify_insert_search_start ()  [inline], [protected]
```
Notifies a search started.

notify_inserted()

```
void __gnu_pbds::sample_resize_policy::notify_inserted (
            size_type num_e)  [inline], [protected]
```
Notifies an element was inserted.

notify_resized()

```
void __gnu_pbds::sample_resize_policy::notify_resized (
            size_type new_size)  [protected]
```
Notifies the table was resized to new_size.

sample_range_hashing()

```
__gnu_pbds::sample_resize_policy::sample_range_hashing (
            const sample_resize_policy & other)
```
Copy constructor.
References sample_resize_policy().

swap()

```
void __gnu_pbds::sample_resize_policy::swap (
            sample_resize_policy & other)  [inline]
```
Swaps content.
References sample_resize_policy().
The documentation for this class was generated from the following file:

- sample_resize_policy.hpp

5.867 __gnu_pbds::sample_resize_trigger Class Reference

`#include <sample_resize_trigger.hpp>`

Public Types

- typedef std::size_t size_type

Public Member Functions

- sample_resize_trigger ()
- sample_range_hashing (const sample_resize_trigger &)
- void swap (sample_resize_trigger &)

Protected Member Functions

- bool is_grow_needed (size_type size, size_type num_entries) const
- bool is_resize_needed () const
- void notify_cleared ()
- void notify_erase_search_collision ()
- void notify_erase_search_end ()
- void notify_erase_search_start ()
- void notify_erased (size_type num_entries)
- void notify_externally_resized (size_type new_size)
- void notify_find_search_collision ()
- void notify_find_search_end ()
- void notify_find_search_start ()
- void notify_insert_search_collision ()
- void notify_insert_search_end ()
- void notify_insert_search_start ()
- void notify_inserted (size_type num_entries)
- void notify_resized (size_type new_size)

5.867.1 Detailed Description

A sample resize trigger policy.

5.867.2 Member Typedef Documentation

size_type

`typedef std::size_t __gnu_pbds::sample_resize_trigger::size_type`
Size type.

5.867.3 Constructor & Destructor Documentation

sample_resize_trigger()

`__gnu_pbds::sample_resize_trigger::sample_resize_trigger ()`
Default constructor.
Referenced by sample_range_hashing(), and swap().

5.867.4 Member Function Documentation

is_grow_needed()

```
bool __gnu_pbds::sample_resize_trigger::is_grow_needed (
            size_type size,
            size_type num_entries) const  [inline], [protected]
```

Queries whether a grow is needed.

is_resize_needed()

```
bool __gnu_pbds::sample_resize_trigger::is_resize_needed () const  [inline], [protected]
```

Queries whether a resize is needed.

notify_cleared()

```
void __gnu_pbds::sample_resize_trigger::notify_cleared ()  [protected]
```

Notifies the table was cleared.

notify_erase_search_collision()

```
void __gnu_pbds::sample_resize_trigger::notify_erase_search_collision ()  [inline], [protected]
```

Notifies a search encountered a collision.

notify_erase_search_end()

```
void __gnu_pbds::sample_resize_trigger::notify_erase_search_end ()  [inline], [protected]
```

Notifies a search ended.

notify_erase_search_start()

```
void __gnu_pbds::sample_resize_trigger::notify_erase_search_start ()  [inline], [protected]
```

Notifies a search started.

notify_erased()

```
void __gnu_pbds::sample_resize_trigger::notify_erased (
            size_type num_entries)  [inline], [protected]
```

Notifies an element was erased.

notify_externally_resized()

```
void __gnu_pbds::sample_resize_trigger::notify_externally_resized (
            size_type new_size)  [protected]
```

Notifies the table was resized externally.

notify_find_search_collision()

```
void __gnu_pbds::sample_resize_trigger::notify_find_search_collision ()  [inline], [protected]
```

Notifies a search encountered a collision.

notify_find_search_end()

```
void __gnu_pbds::sample_resize_trigger::notify_find_search_end ()  [inline], [protected]
```

Notifies a search ended.

notify_find_search_start()

```
void __gnu_pbds::sample_resize_trigger::notify_find_search_start ()  [inline], [protected]
```
Notifies a search started.

notify_insert_search_collision()

```
void __gnu_pbds::sample_resize_trigger::notify_insert_search_collision ()  [inline], [protected]
```
Notifies a search encountered a collision.

notify_insert_search_end()

```
void __gnu_pbds::sample_resize_trigger::notify_insert_search_end ()  [inline], [protected]
```
Notifies a search ended.

notify_insert_search_start()

```
void __gnu_pbds::sample_resize_trigger::notify_insert_search_start ()  [inline], [protected]
```
Notifies a search started.

notify_inserted()

```
void __gnu_pbds::sample_resize_trigger::notify_inserted (
            size_type num_entries)  [inline], [protected]
```
Notifies an element was inserted. the total number of entries in the table is num_entries.

notify_resized()

```
void __gnu_pbds::sample_resize_trigger::notify_resized (
            size_type new_size)  [protected]
```
Notifies the table was resized as a result of this object's signifying that a resize is needed.

sample_range_hashing()

```
__gnu_pbds::sample_resize_trigger::sample_range_hashing (
            const sample_resize_trigger & )
```
Copy constructor.
References sample_resize_trigger().

swap()

```
void __gnu_pbds::sample_resize_trigger::swap (
            sample_resize_trigger & )  [inline]
```
Swaps content.
References sample_resize_trigger().
The documentation for this class was generated from the following file:

- sample_resize_trigger.hpp

5.868 __gnu_pbds::sample_size_policy Class Reference

```
#include <sample_size_policy.hpp>
```

Public Types

- typedef std::size_t size_type

Public Member Functions

- sample_size_policy ()
- sample_range_hashing (const sample_size_policy &)
- void swap (sample_size_policy &other)

Protected Member Functions

- size_type get_nearest_larger_size (size_type size) const
- size_type get_nearest_smaller_size (size_type size) const

5.868.1 Detailed Description

A sample size policy.

5.868.2 Member Typedef Documentation

size_type

```
typedef std::size_t __gnu_pbds::sample_size_policy::size_type
```

Size type.

5.868.3 Constructor & Destructor Documentation

sample_size_policy()

```
__gnu_pbds::sample_size_policy::sample_size_policy ()
```

Default constructor.
Referenced by sample_range_hashing(), and swap().

5.868.4 Member Function Documentation

get_nearest_larger_size()

```
size_type __gnu_pbds::sample_size_policy::get_nearest_larger_size (
            size_type size) const  [inline], [protected]
```

Given a __size size, returns a __size that is larger.

get_nearest_smaller_size()

```
size_type __gnu_pbds::sample_size_policy::get_nearest_smaller_size (
            size_type size) const  [inline], [protected]
```

Given a __size size, returns a __size that is smaller.

sample_range_hashing()

```
__gnu_pbds::sample_size_policy::sample_range_hashing (
            const sample_size_policy & )
```

Copy constructor.
References sample_size_policy().

swap()

```
void __gnu_pbds::sample_size_policy::swap (
            sample_size_policy & other)  [inline]
```

Swaps content.
References sample_size_policy().

The documentation for this class was generated from the following file:

- sample_size_policy.hpp

5.869 __gnu_pbds::sample_tree_node_update< Const_Node_Iter, Node_Iter, Cmp_Fn, _Alloc > Class Template Reference

`#include <sample_tree_node_update.hpp>`

5.869.1 Detailed Description

template<typename Const_Node_Iter, typename Node_Iter, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::sample_tree_node_update< Const_Node_Iter, Node_Iter, Cmp_Fn, _Alloc >

A sample node updator.
The documentation for this class was generated from the following file:

- sample_tree_node_update.hpp

5.870 __gnu_pbds::sample_trie_access_traits Struct Reference

`#include <sample_trie_access_traits.hpp>`

Public Types

- enum { **max_size** }
- typedef std::string::const_iterator **const_iterator**
- typedef char e_type
- typedef rebind_traits< _Alloc, key_type >::const_reference **key_const_reference**
- typedef std::string **key_type**
- typedef std::size_t **size_type**

Static Public Member Functions

- static const_iterator begin (key_const_reference)
- static size_type e_pos (e_type)
- static const_iterator end (key_const_reference)

5.870.1 Detailed Description

A sample trie element access traits.

5.870.2 Member Typedef Documentation

e_type

```
typedef char __gnu_pbds::sample_trie_access_traits::e_type
```
Element type.

5.870.3 Member Function Documentation

begin()

```
static const_iterator __gnu_pbds::sample_trie_access_traits::begin (
            key_const_reference )  [inline], [static]
```
Returns a const_iterator to the first element of r_key.

e_pos()

```
static size_type __gnu_pbds::sample_trie_access_traits::e_pos (
            e_type )  [inline], [static]
```

Maps an element to a position.

end()

```
static const_iterator __gnu_pbds::sample_trie_access_traits::end (
            key_const_reference )  [inline], [static]
```

Returns a const_iterator to the after-last element of r_key.
The documentation for this struct was generated from the following file:

- sample_trie_access_traits.hpp

5.871 __gnu_pbds::sample_trie_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc > Class Template Reference

```
#include <sample_trie_node_update.hpp>
```

Public Types

- typedef std::size_t **metadata_type**

Protected Member Functions

- sample_trie_node_update ()
- void operator() (node_iterator, node_const_iterator) const

5.871.1 Detailed Description

template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
class __gnu_pbds::sample_trie_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >

A sample node updator.

5.871.2 Constructor & Destructor Documentation

sample_trie_node_update()

```
template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
__gnu_pbds::sample_trie_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::sample_trie_node_↩
update ()  [protected]
```

Default constructor.

5.871.3 Member Function Documentation

operator()()

```
template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
void __gnu_pbds::sample_trie_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::operator() (
            node_iterator ,
            node_const_iterator ) const  [inline], [protected]
```

Updates the rank of a node through a node_iterator node_it; end_nd_it is the end node iterator.
The documentation for this class was generated from the following file:

- sample_trie_node_update.hpp

5.872 __gnu_pbds::sample_update_policy Struct Reference

`#include <sample_update_policy.hpp>`

Public Member Functions

- sample_update_policy ()
- sample_update_policy (const sample_update_policy &)
- void swap (sample_update_policy &other)

Protected Types

- typedef some_metadata_type metadata_type

Protected Member Functions

- metadata_type operator() () const
- bool operator() (metadata_reference) const

5.872.1 Detailed Description

A sample list-update policy.

5.872.2 Member Typedef Documentation

metadata_type

```
typedef some_metadata_type __gnu_pbds::sample_update_policy::metadata_type  [protected]
```

Metadata on which this functor operates.

5.872.3 Constructor & Destructor Documentation

sample_update_policy() [1/2]

```
__gnu_pbds::sample_update_policy::sample_update_policy ()
```

Default constructor.
Referenced by sample_update_policy(), and swap().

sample_update_policy() [2/2]

```
__gnu_pbds::sample_update_policy::sample_update_policy (
            const sample_update_policy & )
```

Copy constructor.
References sample_update_policy().

5.872.4 Member Function Documentation

operator()() [1/2]

```
metadata_type __gnu_pbds::sample_update_policy::operator() () const  [protected]
```

Creates a metadata object.

operator()() [2/2]

```
bool __gnu_pbds::sample_update_policy::operator() (
            metadata_reference ) const  [protected]
```

Decides whether a metadata object should be moved to the front of the list. A list-update based containers object will call this method to decide whether to move a node to the front of the list. The method shoule return true if the node should be moved to the front of the list.

swap()

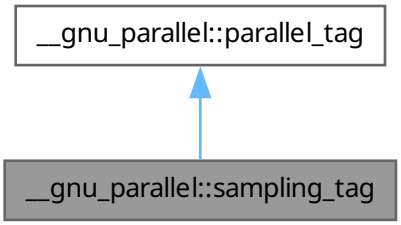
```
void __gnu_pbds::sample_update_policy::swap (
            sample_update_policy & other)  [inline]
```

Swaps content.
References sample_update_policy().
The documentation for this struct was generated from the following file:

- sample_update_policy.hpp

5.873 __gnu_parallel::sampling_tag Struct Reference

`#include <tags.h>`
Inheritance diagram for __gnu_parallel::sampling_tag:



Public Member Functions

- **sampling_tag** (_ThreadIndex __num_threads)
- _ThreadIndex __get_num_threads ()
- void set_num_threads (_ThreadIndex __num_threads)

5.873.1 Detailed Description

Forces parallel merging with exact splitting, at compile time.

5.873.2 Member Function Documentation

__get_num_threads()

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ()  [inline], [inherited]
```

Find out desired number of threads.

Returns

Desired number of threads.

Referenced by __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), and __gnu_parallel::__parallel_sort(

set_num_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
            _ThreadIndex __num_threads)  [inline], [inherited]
```

Set the desired number of threads.

Parameters

__num_threads	Desired number of threads.

The documentation for this struct was generated from the following file:

- tags.h

5.874 std::scoped_allocator_adaptor< _OuterAlloc, _InnerAllocs > Class Template Reference

```
#include <scoped_allocator>
```

Public Types

- typedef __traits::const_pointer **const_pointer**
- typedef __traits::const_void_pointer **const_void_pointer**
- typedef __traits::difference_type **difference_type**
- typedef __inner_type::__type **inner_allocator_type**
- typedef __and_< typename __traits::is_always_equal, typenameallocator_traits< _InnerAllocs >::is_always_equal... >::type **is_always_equal**
- typedef _OuterAlloc **outer_allocator_type**
- typedef __traits::pointer **pointer**
- typedef __or_< typename __traits::propagate_on_container_copy_assignment, typenameallocator_traits< _InnerAllocs >::propagate_on_container_copy_assignment... >::type **propagate_on_container_copy_assignment**
- typedef __or_< typename __traits::propagate_on_container_move_assignment, typenameallocator_traits< _InnerAllocs >::propagate_on_container_move_assignment... >::type **propagate_on_container_move_assignment**
- typedef __or_< typename __traits::propagate_on_container_swap, typenameallocator_traits< _InnerAllocs >::propagate_on_container_swap... >::type **propagate_on_container_swap**
- typedef __traits::size_type **size_type**
- typedef __traits::value_type **value_type**
- typedef __traits::void_pointer **void_pointer**

Public Member Functions

- template<typename _Outer2, typename = _Constructible<_Outer2>>
  scoped_allocator_adaptor (_Outer2 &&__outer, const _InnerAllocs &... __inner) noexcept
- **scoped_allocator_adaptor** (const scoped_allocator_adaptor &__other) noexcept
- template<typename _Outer2, typename = _Constructible<const _Outer2&>>
  scoped_allocator_adaptor (const scoped_allocator_adaptor< _Outer2, _InnerAllocs... > &__other) noexcept

- **scoped_allocator_adaptor** (scoped_allocator_adaptor &&__other) noexcept
- template<typename _Outer2, typename = _Constructible<_Outer2>>
  scoped_allocator_adaptor (scoped_allocator_adaptor< _Outer2, _InnerAllocs... > &&__other) noexcept
- pointer **allocate** (size_type __n)
- pointer **allocate** (size_type __n, const_void_pointer __hint)
- template<typename _Tp, typename... _Args>
  void **construct** (_Tp *__p, _Args &&... __args)
- void **deallocate** (pointer __p, size_type __n) noexcept
- template<typename _Tp>
  void **destroy** (_Tp *__p)
- const inner_allocator_type & **inner_allocator** () const noexcept
- inner_allocator_type & **inner_allocator** () noexcept
- size_type **max_size** () const
- scoped_allocator_adaptor & **operator=** (const scoped_allocator_adaptor &)=default
- scoped_allocator_adaptor & **operator=** (scoped_allocator_adaptor &&)=default
- const outer_allocator_type & **outer_allocator** () const noexcept
- outer_allocator_type & **outer_allocator** () noexcept
- scoped_allocator_adaptor **select_on_container_copy_construction** () const

Friends

- template<typename...>
  struct **__inner_type_impl**
- template<typename _OutA1, typename _OutA2, typename... _InA>
  bool **operator==** (const scoped_allocator_adaptor< _OutA1, _InA... > &__a, const scoped_allocator_adaptor< _OutA2, _InA... > &__b) noexcept

Related Symbols

(Note that these are not member symbols.)

- template<typename _OutA1, typename _OutA2, typename... _InA>
  bool **operator==** (const scoped_allocator_adaptor< _OutA1, _InA... > &__a, const scoped_allocator_adaptor< _OutA2, _InA... > &__b) noexcept

5.874.1 Detailed Description

template<typename _OuterAlloc, typename... _InnerAllocs>
class std::scoped_allocator_adaptor< _OuterAlloc, _InnerAllocs >

An adaptor to recursively pass an allocator to the objects it constructs.
The documentation for this class was generated from the following file:

- scoped_allocator

5.875 std::seed_seq Class Reference

`#include <random>`

Public Types

- typedef uint_least32_t result_type

Public Member Functions

- seed_seq () noexcept
- template<typename _InputIterator>
  seed_seq (_InputIterator __begin, _InputIterator __end)
- **seed_seq** (const seed_seq &)=delete
- template<typename _IntType, typename = _Require<is_integral<_IntType>>>
  seed_seq (std::initializer_list< _IntType > __il)
- template<typename _RandomAccessIterator>
  void **generate** (_RandomAccessIterator __begin, _RandomAccessIterator __end)
- seed_seq & **operator=** (const seed_seq &)=delete
- template<typename _OutputIterator>
  void **param** (_OutputIterator __dest) const
- size_t **size** () const noexcept

5.875.1 Detailed Description

The seed_seq class generates sequences of seeds for random number generators.

Since

C++11

5.875.2 Member Typedef Documentation

result_type

```
typedef uint_least32_t std::seed_seq::result_type
```

The type of the seed vales.

5.875.3 Constructor & Destructor Documentation

seed_seq()

```
std::seed_seq::seed_seq ()  [inline], [noexcept]
```

Default constructor.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

5.876 __gnu_cxx::select1st< _Pair > Struct Template Reference

```
#include <functional>
```

Public Types

- typedef _Pair argument_type
- typedef _Pair::first_type result_type

Public Member Functions

- _Pair::first_type & **operator()** (_Pair &__x) const
- template<typename _Pair2>
  _Pair2::first_type & **operator()** (_Pair2 &__x) const
- const _Pair::first_type & **operator()** (const _Pair &__x) const
- template<typename _Pair2>
  const _Pair2::first_type & **operator()** (const _Pair2 &__x) const

5.876.1 Detailed Description

template<class _Pair>
struct __gnu_cxx::select1st< _Pair >

An SGI extension .

5.876.2 Member Typedef Documentation

argument_type

```
typedef _Pair std::unary_function< _Pair, _Pair::first_type >::argument_type  [inherited]
```

`argument_type` is the type of the argument

result_type

```
typedef _Pair::first_type std::unary_function< _Pair, _Pair::first_type >::result_type  [inherited]
```

`result_type` is the return type
The documentation for this struct was generated from the following file:

- ext/functional

5.877 __gnu_cxx::select2nd< _Pair > Struct Template Reference

`#include <functional>`

Public Types

- typedef _Pair argument_type
- typedef _Pair::second_type result_type

Public Member Functions

- _Pair::second_type & **operator()** (_Pair &__x) const
- const _Pair::second_type & **operator()** (const _Pair &__x) const

5.877.1 Detailed Description

template<class _Pair>
struct __gnu_cxx::select2nd< _Pair >

An SGI extension .

5.877.2 Member Typedef Documentation

argument_type

```
typedef _Pair std::unary_function< _Pair, _Pair::second_type >::argument_type  [inherited]
```

`argument_type` is the type of the argument

result_type

```
typedef _Pair::second_type std::unary_function< _Pair, _Pair::second_type >::result_type  [inherited]
```

`result_type` is the return type
The documentation for this struct was generated from the following file:

- ext/functional

5.878 __gnu_pbds::detail::select_value_type< Key, Mapped > Struct Template Reference

`#include <types_traits.hpp>`

Public Types

- typedef std::pair< const Key, Mapped > **type**

5.878.1 Detailed Description

template<typename Key, typename Mapped>
struct __gnu_pbds::detail::select_value_type< Key, Mapped >

Choose value_type to be a key/value pair or just a key.
The documentation for this struct was generated from the following file:

- types_traits.hpp

5.879 __gnu_pbds::detail::select_value_type< Key, null_type > Struct Template Reference

`#include <types_traits.hpp>`

Public Types

- typedef std::pair< const Key, null_type > **type**
- typedef Key **type**

5.879.1 Detailed Description

template<typename Key>
struct __gnu_pbds::detail::select_value_type< Key, null_type >

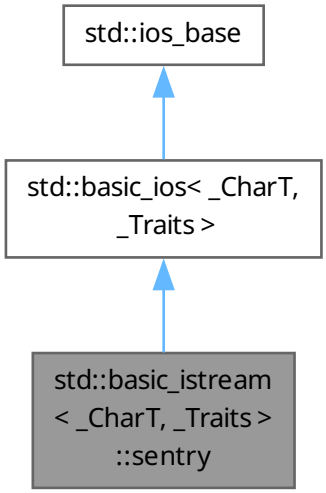
Specialization for sets where the key is the value_type.
The documentation for this struct was generated from the following file:

- types_traits.hpp

5.880 std::basic_istream< _CharT, _Traits >::sentry Class Reference

`#include <istream>`

Inheritance diagram for std::basic_istream< _CharT, _Traits >::sentry:



Public Types

- typedef ctype< _CharT > **__ctype_type**
- typedef __istream_type::__ctype_type **__ctype_type**
- typedef _Traits::int_type **__int_type**
- typedef basic_ios< _CharT, _Traits > **__ios_type**
- typedef basic_istream< _CharT, _Traits > **__istream_type**
- typedef basic_istream< _CharT, _Traits > **__istream_type**
- typedef num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > **__num_get_type**
- typedef basic_streambuf< _CharT, _Traits > **__streambuf_type**
- typedef basic_streambuf< _CharT, _Traits > **__streambuf_type**
- typedef _CharT **char_type**
- enum event
- typedef void(* event_callback) (event __e, ios_base &__b, int __i)
- typedef _Traits::int_type **int_type**
- typedef _Ios_Iostate iostate
- typedef _Traits::off_type **off_type**
- typedef _Ios_Openmode openmode
- typedef _Traits::pos_type **pos_type**
- typedef _Ios_Seekdir seekdir
- typedef _Traits **traits_type**
- typedef _Traits traits_type

- typedef num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > __num_put_type

Public Member Functions

- sentry (basic_istream< _CharT, _Traits > &__is, bool __noskipws=false)
- virtual ~basic_istream ()
- basic_istream< _CharT, _Traits > & **_M_extract** (_ValueT &__v)
- basic_istream< _CharT, _Traits > & **_M_extract** (_ValueT &__v)
- const locale & _M_getloc () const
- void **_M_setstate** (iostate __state)
- bool bad () const
- basic_istream (__streambuf_type *__sb)
- void clear (iostate __state=goodbit)
- basic_ios & copyfmt (const basic_ios &__rhs)
- bool eof () const
- iostate exceptions () const
- void exceptions (iostate __except)
- bool fail () const
- char_type fill () const
- char_type fill (char_type __ch)
- fmtflags flags () const
- fmtflags flags (fmtflags __fmtfl)
- streamsize gcount () const
- basic_istream< char > & getline (char_type *__s, streamsize __n, char_type __delim)
- basic_istream< char > & getline (char_type *__s, streamsize __n, char_type __delim)
- basic_istream< wchar_t > & **getline** (char_type *__s, streamsize __n, char_type __delim)
- basic_istream< wchar_t > & **getline** (char_type *__s, streamsize __n, char_type __delim)
- locale getloc () const
- bool good () const
- basic_istream< char > & **ignore** (streamsize __n)
- basic_istream< char > & **ignore** (streamsize __n)
- basic_istream< wchar_t > & **ignore** (streamsize __n)
- basic_istream< wchar_t > & **ignore** (streamsize __n)
- basic_istream< char > & **ignore** (streamsize __n, int_type __delim)
- basic_istream< char > & **ignore** (streamsize __n, int_type __delim)
- basic_istream< wchar_t > & **ignore** (streamsize __n, int_type __delim)
- basic_istream< wchar_t > & **ignore** (streamsize __n, int_type __delim)
- locale imbue (const locale &__loc)
- long & iword (int __ix)
- char narrow (char_type __c, char __dfault) const
- operator bool () const
- __istream_type & operator>> (__streambuf_type *__sb)
- __istream_type & operator>> (void *&__p)
- streamsize precision () const
- streamsize precision (streamsize __prec)
- void *& pword (int __ix)
- basic_streambuf< _CharT, _Traits > * rdbuf () const
- basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > *__sb)
- iostate rdstate () const
- void register_callback (event_callback __fn, int __index)
- fmtflags setf (fmtflags __fmtfl)
- fmtflags setf (fmtflags __fmtfl, fmtflags __mask)
- void setstate (iostate __state)

- basic_ostream< _CharT, _Traits > * tie () const
- basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > *__tiestr)
- void unsetf (fmtflags __mask)
- char_type widen (char __c) const
- streamsize width () const
- streamsize width (streamsize __wide)

- __istream_type & operator>> (__istream_type &(*__pf)(__istream_type &))
- __istream_type & operator>> (__ios_type &(*__pf)(__ios_type &))
- __istream_type & operator>> (ios_base &(*__pf)(ios_base &))

Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type std::basic_istream::sentry with the second argument (noskipws) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, ios_base::badbit will be turned on in the stream's error state (without causing an ios_base::failure to be thrown) and the original exception will be rethrown if badbit is set in the exceptions mask.

- __istream_type & operator>> (bool &__n)
- __istream_type & operator>> (short &__n)
- __istream_type & operator>> (unsigned short &__n)
- __istream_type & operator>> (int &__n)
- __istream_type & operator>> (unsigned int &__n)
- __istream_type & operator>> (long &__n)
- __istream_type & operator>> (unsigned long &__n)
- __istream_type & operator>> (long long &__n)
- __istream_type & operator>> (unsigned long long &__n)

- __istream_type & operator>> (float &__f)
- __istream_type & operator>> (double &__f)
- __istream_type & operator>> (long double &__f)

Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type std::basic_istream::sentry with the second argument (noskipws) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by gcount().

If an exception is thrown during extraction, ios_base::badbit will be turned on in the stream's error state (without causing an ios_base::failure to be thrown) and the original exception will be rethrown if badbit is set in the exceptions mask.

- int_type get ()
- __istream_type & get (char_type &__c)
- __istream_type & get (char_type *__s, streamsize __n, char_type __delim)
- __istream_type & get (char_type *__s, streamsize __n)
- __istream_type & get (__streambuf_type &__sb, char_type __delim)
- __istream_type & get (__streambuf_type &__sb)
- __istream_type & getline (char_type *__s, streamsize __n, char_type __delim)
- __istream_type & getline (char_type *__s, streamsize __n)
- __istream_type & ignore (streamsize __n, int_type __delim)

- __istream_type & ignore (streamsize __n)
- __istream_type & ignore ()
- int_type peek ()
- __istream_type & read (char_type *__s, streamsize __n)
- streamsize readsome (char_type *__s, streamsize __n)
- __istream_type & putback (char_type __c)
- __istream_type & unget ()
- int sync ()
- pos_type tellg ()
- __istream_type & seekg (pos_type)
- __istream_type & seekg (off_type, ios_base::seekdir)

- operator bool () const
- bool operator! () const

Static Public Member Functions

- static bool sync_with_stdio (bool __sync=true)
- static int xalloc () throw ()

Public Attributes

- class __attribute((__abi_tag__("cxx11"))) failure typedef _Ios_Fmtflags fmtflags

Static Public Attributes

- static const openmode **__noreplace**
- static const fmtflags adjustfield
- static const openmode app
- static const openmode ate
- static const iostate badbit
- static const fmtflags basefield
- static const seekdir beg
- static const openmode binary
- static const fmtflags boolalpha
- static const seekdir cur
- static const fmtflags dec
- static const seekdir end
- static const iostate eofbit
- static const iostate failbit
- static const fmtflags fixed
- static const fmtflags floatfield
- static const iostate goodbit
- static const fmtflags hex
- static const openmode in
- static const fmtflags internal
- static const fmtflags left
- static const fmtflags oct
- static const openmode out
- static const fmtflags right
- static const fmtflags scientific
- static const fmtflags showbase
- static const fmtflags showpoint

- static const fmtflags showpos
- static const fmtflags skipws
- static const openmode trunc
- static const fmtflags unitbuf
- static const fmtflags uppercase

Protected Types

- enum

Protected Member Functions

- void **_M_cache_locale** (const locale &__loc)
- void **_M_call_callbacks** (event __ev) throw ()
- void **_M_call_callbacks** (event __ev) throw ()
- void **_M_dispose_callbacks** (void) throw ()
- __istream_type & **_M_extract** (_ValueT &__v)
- _Words & **_M_grow_words** (int __index, bool __iword)
- void **_M_init** () throw ()
- void **_M_move** (ios_base &) noexcept
- void **_M_swap** (ios_base &__rhs) noexcept
- **basic_istream** ()
- **basic_istream** (basic_istream &&__rhs)
- **basic_istream** (const basic_istream &)=delete
- void init (basic_streambuf< _CharT, _Traits > *__sb)
- void **move** (basic_ios &&__rhs)
- void **move** (basic_ios &__rhs)
- basic_istream & **operator=** (basic_istream &&__rhs)
- basic_istream & **operator=** (const basic_istream &)=delete
- void **set_rdbuf** (basic_streambuf< _CharT, _Traits > *__sb)
- void **swap** (basic_ios &__rhs) noexcept
- void **swap** (basic_istream &__rhs)

Protected Attributes

- _Callback_list * **_M_callbacks**
- const __ctype_type * **_M_ctype**
- iostate **_M_exception**
- char_type **_M_fill**
- bool **_M_fill_init**
- fmtflags **_M_flags**
- streamsize _M_gcount
- locale **_M_ios_locale**
- _Words **_M_local_word** [_S_local_word_size]
- const __num_get_type * **_M_num_get**
- const __num_put_type * **_M_num_put**
- streamsize **_M_precision**
- basic_streambuf< _CharT, _Traits > * **_M_streambuf**
- iostate **_M_streambuf_state**
- basic_ostream< _CharT, _Traits > * **_M_tie**
- streamsize **_M_width**
- _Words * **_M_word**
- int **_M_word_size**
- _Words **_M_word_zero**

5.880.1 Detailed Description

template<typename _CharT, typename _Traits>
class std::basic_istream< _CharT, _Traits >::sentry

Performs setup work for input streams.
Objects of this class are created before all of the standard extractors are run. It is responsible for *exception-safe prefix and suffix operations,* although only prefix actions are currently required by the standard.

5.880.2 Member Typedef Documentation

__num_put_type

```
typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios< _CharT, _Traits
>::__num_put_type
```

These are non-standard types.

event_callback

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i)
```

The type of an event callback function.

Parameters

_e	One of the members of the event enum.
_b	Reference to the ios_base object.
_i	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several ios_base and basic_ios functions, specifically imbue(), copyfmt(), and ~ios().

iostate

```
typedef _Ios_Iostate std::ios_base::iostate
```

This is a bitmask type.
_Ios_Iostate is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit
- eofbit
- failbit
- goodbit

openmode

```
typedef _Ios_Openmode std::ios_base::openmode
```

This is a bitmask type.
_Ios_Openmode is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type openmode are:

- app

- ate
- binary
- in
- out
- trunc

seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir
```
This is an enumerated type.
_Ios_Seekdir is implementation-defined. Defined values of type seekdir are:

- beg
- cur, equivalent to SEEK_CUR in the C standard library.
- end, equivalent to SEEK_END in the C standard library.

traits_type

```
template<typename _CharT, typename _Traits>
typedef _Traits std::basic_istream< _CharT, _Traits >::sentry::traits_type
```
Easy access to dependent types.

5.880.3 Member Enumeration Documentation

event

```
enum std::ios_base::event
```
The set of events that may be passed to an event callback.
erase_event is used during ~ios() and copyfmt(). imbue_event is used during imbue(). copyfmt_event is used during copyfmt().

5.880.4 Constructor & Destructor Documentation

sentry()

```
template<typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits >::sentry::sentry (
            basic_istream< _CharT, _Traits > & __is,
            bool __noskipws = false)  [explicit]
```
The constructor performs all the work.

Parameters

__is	The input stream to guard.
__noskipws	Whether to consume whitespace or not.

If the stream state is good (*__is.good()* is true), then the following actions are performed, otherwise the sentry state is false (*not okay*) and failbit is set in the stream state.
The sentry's preparatory actions are:

1. if the stream is tied to an output stream, is.tie()->flush() is called to synchronize the output sequence

2. if *__noskipws* is false, and `ios_base::skipws` is set in `is.flags()`, the sentry extracts and discards whitespace characters from the stream. The currently imbued locale is used to determine whether each character is whitespace.

If the stream state is still good, then the sentry state becomes true (*okay*).
References std::ios_base::badbit, basic_istream(), std::basic_ios< _CharT, _Traits >::good(), and std::ios_base::goodbit.

~basic_istream()

```
virtual std::basic_istream< _CharT, _Traits >::~basic_istream ()  [inline], [virtual]
```
Base destructor.
This does very little apart from providing a virtual base dtor.

5.880.5 Member Function Documentation

_M_getloc()

```
const locale & std::ios_base::_M_getloc () const  [inline]
```
Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

bad()

```
bool std::basic_ios< _CharT, _Traits >::bad () const  [inline], [nodiscard]
```
Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.

basic_istream()

```
std::basic_istream< _CharT, _Traits >::basic_istream (
            __streambuf_type * __sb)  [inline], [explicit]
```
Base constructor.
This ctor is almost never called by the user directly, rather from derived classes' initialization lists, which pass a pointer to their own stream buffer.
Referenced by sentry().

clear()

```
void std::basic_ios< _CharT, _Traits >::clear (
            iostate __state = goodbit)
```
[Re]sets the error state.

Parameters

__state	The new state flag(s) to set.

See std::ios_base::iostate for the possible bit values. Most users will not need to pass an argument.

copyfmt()

```
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
            const basic_ios & __rhs)
```

Copies fields of __rhs into this.

Parameters

__rhs	The source values for the copies.

Returns

Reference to this object.

All fields of __rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase_event. After copying, each (new) callback is invoked with copyfmt_event. The final step is to copy exceptions().

eof()

```
bool std::basic_ios< _CharT, _Traits >::eof () const  [inline], [nodiscard]
```

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

exceptions() [1/2]

```
iostate std::basic_ios< _CharT, _Traits >::exceptions () const  [inline], [nodiscard]
```

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.

exceptions() [2/2]

```
void std::basic_ios< _CharT, _Traits >::exceptions (
            iostate __except)  [inline]
```

Throwing exceptions on errors.

Parameters

__except	The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type std::ios_base::failure is thrown.
If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
```

```
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr « "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr « "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

fail()

```
bool std::basic_ios< _CharT, _Traits >::fail () const  [inline], [nodiscard]
```
Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.

fill() [1/2]

```
char_type std::basic_ios< _CharT, _Traits >::fill () const  [inline], [nodiscard]
```
Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.

fill() [2/2]

```
char_type std::basic_ios< _CharT, _Traits >::fill (
            char_type __ch)  [inline]
```
Sets a new *empty* character.

Parameters

__ch	The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

flags() [1/2]

```
fmtflags std::ios_base::flags () const  [inline], [nodiscard]
```
Access to format flags.

Returns

The format control flags for both input and output.

flags() [2/2]

```
fmtflags std::ios_base::flags (
            fmtflags __fmtfl)  [inline]
```

Setting new format flags all at once.

Parameters

__fmtfl	The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *__fmtfl*.

gcount()

```
streamsize std::basic_istream< _CharT, _Traits >::gcount () const  [inline]
```

Character counting.

Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

get() [1/6]

```
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::get (
            void )
```

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

get() [2/6]

```
__istream_type & std::basic_istream< _CharT, _Traits >::get (
            __streambuf_type & __sb)  [inline]
```

Extraction into another streambuf.

Parameters

__sb	A streambuf in which to store data.

Returns

*this

Returns `get`(__sb,widen('\n')).

get() [3/6]

```
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
            __streambuf_type & __sb,
            char_type __delim)
```

Extraction into another streambuf.

Parameters

__sb	A streambuf in which to store data.
__delim	A "stop" character.

Returns

*this

Characters are extracted and inserted into *__sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals *__delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.

get() [4/6]

```
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
            char_type & __c)
```

Simple extraction.

Parameters

_↩ _c	The character in which to store data.

Returns

*this

Tries to extract a character and store it in *__c*. If none are available, sets failbit and returns traits::eof().

Note

This function is not overloaded on signed char and unsigned char.

get() [5/6]

```
__istream_type & std::basic_istream< _CharT, _Traits >::get (
            char_type * __s,
            streamsize __n)  [inline]
```

Simple multiple-character extraction.

Parameters

_↩ _s	Pointer to an array.
_↩ _n	Maximum number of characters to store in *s*.

Returns

*this

Returns `get(__s,__n,widen('\n'))`.

get() [6/6]

```
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
            char_type * __s,
            streamsize __n,
            char_type __delim)
```

Simple multiple-character extraction.

Parameters

__s	Pointer to an array.
__n	Maximum number of characters to store in *__s*.
__delim	A "stop" character.

Returns

*this

Characters are extracted and stored into *__s* until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals *__delim*, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.
In any case, a null character is stored into the next location in the array.

Note

This function is not overloaded on signed char and unsigned char.

getline() [1/4]

```
__istream_type & std::basic_istream< _CharT, _Traits >::getline (
            char_type * __s,
            streamsize __n)  [inline]
```

String extraction.

Parameters

_↩ _s	A character array in which to store the data.
_↩ _n	Maximum number of characters to extract.

Returns

*this

Returns `getline`(__s,__n,widen('\n')).

getline() [2/4]

```
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::getline (
            char_type * __s,
            streamsize __n,
            char_type __delim)
```

String extraction.

Parameters

__s	A character array in which to store the data.
__n	Maximum number of characters to extract.
__delim	A "stop" character.

Returns

*this

Extracts and stores characters into *__s* until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the *__s* array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)
In any case, a null character is stored in the next location in the array.

getline() [3/4]

```
basic_istream< char > & std::basic_istream< char >::getline (
            char_type * __s,
            streamsize __n,
            char_type __delim)
```

Explicit specialization declarations, defined in src/istream.cc.

getline() [4/4]

```
basic_istream< char > & std::basic_istream< char >::getline (
            char_type * __s,
            streamsize __n,
            char_type __delim)
```

Explicit specialization declarations, defined in src/istream.cc.

getloc()

```
locale std::ios_base::getloc () const  [inline], [nodiscard]
```

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

good()

```
bool std::basic_ios< _CharT, _Traits >::good () const  [inline], [nodiscard]
```

Fast error checking.

Returns

True if no error flags are set.

A wrapper around rdstate.

ignore() [1/3]

```
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
            void )
```

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

ignore() [2/3]

```
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
            streamsize __n)
```

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

ignore() [3/3]

```
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
            streamsize __n,
            int_type __delim)
```

Discarding characters.

Parameters

__n	Number of characters to discard.
__delim	A "stop" character.

Returns

*this

Extracts characters and throws them away until one of the following happens:

- if *__n* != `std::numeric_limits<int>::max()`, *__n* characters are extracted
- the input sequence reaches end-of-file
- the next character equals *__delim* (in this case, the character is extracted); note that this condition will never occur if *__delim* equals `traits::eof()`.

NB: Provide three overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.

imbue()

```
locale std::basic_ios< _CharT, _Traits >::imbue (
            const locale & __loc)
```

Moves to a new locale.

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.
Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html

init()

```
void std::basic_ios< _CharT, _Traits >::init (
            basic_streambuf< _CharT, _Traits > * __sb)  [protected]
```

All setup is performed here.
This is called from the public constructor. It is not virtual and cannot be redefined.

iword()

```
long & std::ios_base::iword (
            int __ix)  [inline]
```

Access to integer array.

Parameters

__ix	Index into the array.

Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

narrow()

```
char std::basic_ios< _CharT, _Traits >::narrow (
            char_type __c,
            char __dfault) const  [inline]
```

Squeezes characters.

Parameters

__c	The character to narrow.
__dfault	The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.
Returns the result of
`std::use_facet<ctype<char_type> >(getloc()).narrow(c,dfault)`
Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html

operator bool() [1/2]

`std::basic_ios< _CharT, _Traits >::operator bool () const [inline], [explicit], [nodiscard]`
The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

operator bool() [2/2]

```
template<typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits >::sentry::operator bool () const [inline], [explicit]
```
Quick status checking.

Returns

The sentry state.

For ease of use, sentries may be converted to booleans. The return value is that of the sentry state (true == okay).

operator"!()

`bool std::basic_ios< _CharT, _Traits >::operator! () const [inline], [nodiscard]`
The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

operator>>() [1/17]

```
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            __ios_type &(* __pf )(__ios_type &)) [inline]
```
Interface for manipulators.
Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`.
For more information, see the iomanip header.

operator>>() [2/17]

```
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            __istream_type &(* __pf )(__istream_type &)) [inline]
```
Interface for manipulators.
Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`.
For more information, see the iomanip header.

operator>>() [3/17]

```
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
            __streambuf_type * __sb)
```
Extracting into another streambuf.

Parameters

__sb	A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.
If __sb is NULL, the stream will set failbit in its error state.
Characters are extracted from this stream and inserted into the __sb streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.

operator>>() [4/17]

```
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            bool & __n)  [inline]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [5/17]

```
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            double & __f)  [inline]
```

Floating point arithmetic extractors.

Parameters

↩ _↩ ↩ _↩ f	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [6/17]

```
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            float & __f)  [inline]
```

Floating point arithmetic extractors.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [7/17]

```
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
            int & __n)
```

Integer arithmetic extractors.

Parameters

_↩ _*n*	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [8/17]

```
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            ios_base &(* __pf )(ios_base &))  [inline]
```

Interface for manipulators.

Manipulators such as std::ws and std::dec use these functions in constructs like std::cin >> std::ws. For more information, see the iomanip header.

operator>>() [9/17]

```
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            long & __n)  [inline]
```

Integer arithmetic extractors.

Parameters

_↩ _*n*	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [10/17]

```
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            long double & __f)  [inline]
```

Floating point arithmetic extractors.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [11/17]

```
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            long long & __n)  [inline]
```

Integer arithmetic extractors.

Parameters

_↩ *_n*	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [12/17]

```
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
            short & __n)
```

Integer arithmetic extractors.

Parameters

_↩ *_n*	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [13/17]

```
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned int & __n)  [inline]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [14/17]

```
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned long & __n)  [inline]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [15/17]

```
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned long long & __n)  [inline]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [16/17]

```
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned short & __n)  [inline]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [17/17]

```
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            void *& __p)  [inline]
```

Basic arithmetic extractors.

Parameters

_↩ _p	A variable of pointer type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

peek()

```
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::peek (
            void )
```

Looking ahead in the stream.

Returns

The next character, or eof().

If, after constructing the sentry object, good() is false, returns traits::eof(). Otherwise reads but does not extract the next input character.

precision() [1/2]

```
streamsize std::ios_base::precision () const  [inline], [nodiscard]
```

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

precision() [2/2]

```
streamsize std::ios_base::precision (
            streamsize __prec)  [inline]
```

Changing flags.

Parameters

__prec	The new precision value.

Returns

The previous value of precision().

putback()

```
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::putback (
            char_type __c)
```

Unextracting a single character.

Parameters

_↩ _c	The character to push back into the input stream.

Returns

*this

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.
If `rdbuf()` is null or if `sputbackc()` fails, sets badbit in the error state.

Note

This function first clears eofbit. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

pword()

```
void *& std::ios_base::pword (
            int __ix)  [inline]
```

Access to void pointer array.

Parameters

_↩ _ix	Index into the array.

Returns

A reference to a void* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

rdbuf() [1/2]

```
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf () const  [inline],
[nodiscard]
```

Accessing the underlying buffer.

Returns

The current stream buffer.

This does not change the state of the stream.

rdbuf() [2/2]

```
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
            basic_streambuf< _CharT, _Traits > * __sb)
```

Changing the underlying buffer.

Parameters

__sb	The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.
Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream     foo;         // or some other derived type
std::streambuf*  p = .....;

foo.ios::rdbuf(p);            // ios == basic_ios<char>
```

rdstate()

```
iostate std::basic_ios< _CharT, _Traits >::rdstate () const  [inline], [nodiscard]
```

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See std::ios_base::iostate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().

read()

```
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::read (
            char_type * __s,
            streamsize __n)
```

Extraction without delimiters.

Parameters

__s	A character array.
__n	Maximum number of characters to store.

Returns

*this

If the stream state is `good()`, extracts characters and stores them into *__s* until one of the following happens:

- *__n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

Note

This function is not overloaded on signed char and unsigned char.

readsome()

```
streamsize std::basic_istream< _CharT, _Traits >::readsome (
            char_type * __s,
            streamsize __n)
```

Extraction until the buffer is exhausted, but no more.

Parameters

_↩ _s	A character array.
_↩ _n	Maximum number of characters to store.

Returns

The number of characters extracted.

Extracts characters and stores them into __s depending on the number of characters remaining in the streambuf's buffer, `rdbuf()->in_avail()`, called `A` here:

- if `A == -1`, sets eofbit and extracts no characters
- if `A == 0`, extracts no characters
- if `A > 0`, extracts `min(A,n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.

register_callback()

```
void std::ios_base::register_callback (
            event_callback __fn,
            int __index)
```

Add the callback __fn with parameter __index.

Parameters

__fn	The function to add.
__index	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

seekg() [1/2]

```
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
            off_type __off,
            ios_base::seekdir __dir)
```

Changing the current read position.

Parameters

__off	A file offset object.
__dir	The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(__off,__dir)`. If that function fails, sets failbit.

Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

seekg() [2/2]

```
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
            pos_type __pos)
```

Changing the current read position.

Parameters

__pos	A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets failbit.

Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

setf() [1/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl)  [inline]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

setf() [2/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl,
            fmtflags __mask)  [inline]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.
__mask	The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

setstate()

```
void std::basic_ios< _CharT, _Traits >::setstate (
             iostate __state)  [inline]
```

Sets additional flags in the error state.

Parameters

__state	The additional state flag(s) to set.

See std::ios_base::iostate for the possible bit values.

sync()

```
int std::basic_istream< _CharT, _Traits >::sync (
             void )
```

Synchronizing the stream buffer.

Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.
Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit and returns -1.
Otherwise, returns 0.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

sync_with_stdio()

```
static bool std::ios_base::sync_with_stdio (
             bool __sync = true)  [static]
```

Interaction with the standard C I/O objects.

Parameters

__sync	Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary

tellg()

```
basic_istream< _CharT, _Traits >::pos_type std::basic_istream< _CharT, _Traits >::tellg (
            void )
```

Getting the current read position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type`(-1) to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0,cur,in)`.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with putback, unget and seekg, eofbit is not cleared first.

tie() [1/2]

```
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie () const  [inline],
[nodiscard]
```

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

tie() [2/2]

```
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie (
            basic_ostream< _CharT, _Traits > * __tiestr)  [inline]
```

Ties this stream to an output stream.

Parameters

__tiestr	The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

unget()

```
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::unget (
            void )
```

Unextracting the previous character.

Returns

*this

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.
If `rdbuf()` is null or if `sungetc()` fails, sets badbit in the error state.

Note

This function first clears eofbit. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

unsetf()

```
void std::ios_base::unsetf (
            fmtflags __mask)  [inline]
```

Clearing format flags.

Parameters

__mask	The flags to unset.

This function clears *__mask* in the format flags.

widen()

```
char_type std::basic_ios< _CharT, _Traits >::widen (
            char __c) const  [inline]
```

Widens characters.

Parameters

_c	The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.
Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html

width() [1/2]

```
streamsize std::ios_base::width () const  [inline], [nodiscard]
```

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.

width() [2/2]

```
streamsize std::ios_base::width (
            streamsize __wide)  [inline]
```

Changing flags.

Parameters

__wide	The new width value.

Returns

The previous value of width().

xalloc()

```
static int std::ios_base::xalloc () throw ( )   [static]
```
Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the iword and pword functions. The expectation is that an application calls xalloc in order to obtain an index in the iword and pword arrays that can be used without fear of conflict.
The implementation maintains a static variable that is incremented and returned on each invocation. xalloc is guaranteed to return an index that is safe to use in the iword and pword arrays.

5.880.6 Member Data Documentation

_M_gcount

```
streamsize std::basic_istream< _CharT, _Traits >::_M_gcount  [protected]
```
The number of characters extracted in the previous unformatted function; see gcount().

adjustfield

```
const fmtflags std::ios_base::adjustfield  [static]
```
A mask of left|right|internal. Useful for the 2-arg form of `setf`.

app

```
const openmode std::ios_base::app  [static]
```
Seek to end before each write.

ate

```
const openmode std::ios_base::ate  [static]
```
Open and seek to end immediately after opening.

badbit

```
const iostate std::ios_base::badbit  [static]
```
Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

basefield

```
const fmtflags std::ios_base::basefield  [static]
```
A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

beg

```
const seekdir std::ios_base::beg  [static]
```
Request a seek relative to the beginning of the stream.

binary

```
const openmode std::ios_base::binary  [static]
```
Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary.

boolalpha

```
const fmtflags std::ios_base::boolalpha [static]
```
Insert/extract `bool` in alphabetic rather than numeric format.

cur

```
const seekdir std::ios_base::cur [static]
```
Request a seek relative to the current position within the sequence.

dec

```
const fmtflags std::ios_base::dec [static]
```
Converts integer input or generates integer output in decimal base.

end

```
const seekdir std::ios_base::end [static]
```
Request a seek relative to the current end of the sequence.

eofbit

```
const iostate std::ios_base::eofbit [static]
```
Indicates that an input operation reached the end of an input sequence.

failbit

```
const iostate std::ios_base::failbit [static]
```
Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

fixed

```
const fmtflags std::ios_base::fixed [static]
```
Generate floating-point output in fixed-point notation.

floatfield

```
const fmtflags std::ios_base::floatfield [static]
```
A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

fmtflags

```
class __attribute ((__abi_tag__ ("cxx11"))) failure typedef _Ios_Fmtflags std::ios_base::fmtflags
```
These are thrown to indicate problems with io.
27.4.2.1.1 Class ios_base::failure
This is a bitmask type.
`_Ios_Fmtflags` `is` implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type fmtflags are:

- boolalpha
- dec
- fixed
- hex

- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

goodbit

`const iostate std::ios_base::goodbit [static]`
Indicates all is well.

hex

`const fmtflags std::ios_base::hex [static]`
Converts integer input or generates integer output in hexadecimal base.

in

`const openmode std::ios_base::in [static]`
Open for input. Default for `ifstream` and fstream.

internal

`const fmtflags std::ios_base::internal [static]`
Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

left

`const fmtflags std::ios_base::left [static]`
Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

oct

`const fmtflags std::ios_base::oct [static]`
Converts integer input or generates integer output in octal base.

out

```
const openmode std::ios_base::out  [static]
```
Open for output. Default for `ofstream` and fstream.

right

```
const fmtflags std::ios_base::right  [static]
```
Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

scientific

```
const fmtflags std::ios_base::scientific  [static]
```
Generates floating-point output in scientific notation.

showbase

```
const fmtflags std::ios_base::showbase  [static]
```
Generates a prefix indicating the numeric base of generated integer output.

showpoint

```
const fmtflags std::ios_base::showpoint  [static]
```
Generates a decimal-point character unconditionally in generated floating-point output.

showpos

```
const fmtflags std::ios_base::showpos  [static]
```
Generates a + sign in non-negative generated numeric output.

skipws

```
const fmtflags std::ios_base::skipws  [static]
```
Skips leading white space before certain input operations.

trunc

```
const openmode std::ios_base::trunc  [static]
```
Truncate an existing stream when opening. Default for `ofstream`.

unitbuf

```
const fmtflags std::ios_base::unitbuf  [static]
```
Flushes output after each output operation.

uppercase

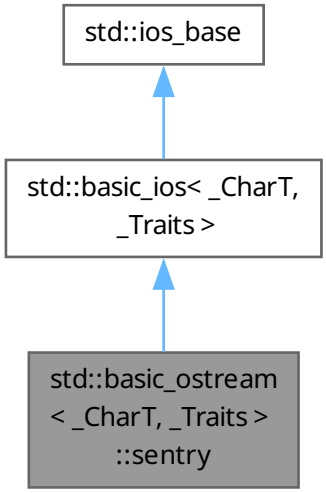
```
const fmtflags std::ios_base::uppercase  [static]
```
Replaces certain lowercase letters with their uppercase equivalents in generated output.
The documentation for this class was generated from the following files:

- istream
- istream.tcc

5.881 std::basic_ostream< _CharT, _Traits >::sentry Class Reference

`#include <ostream.h>`

Inheritance diagram for std::basic_ostream< _CharT, _Traits >::sentry:



Public Types

- typedef ctype< _CharT > **__ctype_type**
- typedef basic_ios< _CharT, _Traits > **__ios_type**
- typedef num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > **__num_put_type**
- typedef basic_ostream< _CharT, _Traits > **__ostream_type**
- typedef basic_streambuf< _CharT, _Traits > **__streambuf_type**
- typedef _CharT **char_type**
- enum event
- typedef void(* event_callback) (event __e, ios_base &__b, int __i)
- typedef _Traits::int_type **int_type**
- typedef _Ios_Iostate iostate
- typedef _Traits::off_type **off_type**
- typedef _Ios_Openmode openmode
- typedef _Traits::pos_type **pos_type**
- typedef _Ios_Seekdir seekdir
- typedef _Traits **traits_type**

- typedef num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > __num_get_type

Public Member Functions

- sentry (basic_ostream< _CharT, _Traits > &__os)
- virtual ~basic_ostream ()

- ~sentry ()
- const locale & _M_getloc () const
- basic_ostream< _CharT, _Traits > & **_M_insert** (_ValueT __v)
- basic_ostream< _CharT, _Traits > & **_M_insert** (_ValueT __v)
- void **_M_setstate** (iostate __state)
- bool bad () const
- basic_ostream (__streambuf_type *__sb)
- void clear (iostate __state=goodbit)
- basic_ios & copyfmt (const basic_ios &__rhs)
- bool eof () const
- iostate exceptions () const
- void exceptions (iostate __except)
- bool fail () const
- char_type fill () const
- char_type fill (char_type __ch)
- fmtflags flags () const
- fmtflags flags (fmtflags __fmtfl)
- __ostream_type & flush ()
- locale getloc () const
- bool good () const
- locale imbue (const locale &__loc)
- long & iword (int __ix)
- char narrow (char_type __c, char __dfault) const
- operator bool () const
- __ostream_type & operator<< (__streambuf_type *__sb)
- __ostream_type & operator<< (const void *__p)
- __ostream_type & **operator**<< (nullptr_t)
- streamsize precision () const
- streamsize precision (streamsize __prec)
- void *& pword (int __ix)
- basic_streambuf< _CharT, _Traits > * rdbuf () const
- basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > *__sb)
- iostate rdstate () const
- void register_callback (event_callback __fn, int __index)
- __ostream_type & seekp (off_type, ios_base::seekdir)
- __ostream_type & seekp (pos_type)
- fmtflags setf (fmtflags __fmtfl)
- fmtflags setf (fmtflags __fmtfl, fmtflags __mask)
- void setstate (iostate __state)
- pos_type tellp ()
- basic_ostream< _CharT, _Traits > * tie () const
- basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > *__tiestr)
- void unsetf (fmtflags __mask)
- char_type widen (char __c) const
- streamsize width () const
- streamsize width (streamsize __wide)

- __ostream_type & operator<< (__ostream_type &(*__pf)(__ostream_type &))
- __ostream_type & operator<< (__ios_type &(*__pf)(__ios_type &))
- __ostream_type & operator<< (ios_base &(*__pf)(ios_base &))

Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type std::basic_ostream::sentry. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, ios_base::badbit will be turned on in the stream's error state without causing an ios_base::failure to be thrown. The original exception will then be rethrown.

- __ostream_type & operator<< (long __n)
- __ostream_type & operator<< (unsigned long __n)
- __ostream_type & operator<< (bool __n)
- __ostream_type & operator<< (short __n)
- __ostream_type & operator<< (unsigned short __n)
- __ostream_type & operator<< (int __n)
- __ostream_type & operator<< (unsigned int __n)
- __ostream_type & operator<< (long long __n)
- __ostream_type & operator<< (unsigned long long __n)

- __ostream_type & operator<< (double __f)
- __ostream_type & operator<< (float __f)
- __ostream_type & operator<< (long double __f)

Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type std::basic_ostream::sentry. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, ios_base::badbit will be turned on in the stream's error state. If badbit is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- __ostream_type & put (char_type __c)
- __ostream_type & write (const char_type *__s, streamsize __n)

- operator bool () const
- bool operator! () const

Static Public Member Functions

- static bool sync_with_stdio (bool __sync=true)
- static int xalloc () throw ()

Public Attributes

- class __attribute((__abi_tag__("cxx11"))) failure typedef _Ios_Fmtflags fmtflags

Static Public Attributes

- static const openmode **__noreplace**
- static const fmtflags adjustfield
- static const openmode app
- static const openmode ate
- static const iostate badbit
- static const fmtflags basefield

- static const seekdir beg
- static const openmode binary
- static const fmtflags boolalpha
- static const seekdir cur
- static const fmtflags dec
- static const seekdir end
- static const iostate eofbit
- static const iostate failbit
- static const fmtflags fixed
- static const fmtflags floatfield
- static const iostate goodbit
- static const fmtflags hex
- static const openmode in
- static const fmtflags internal
- static const fmtflags left
- static const fmtflags oct
- static const openmode out
- static const fmtflags right
- static const fmtflags scientific
- static const fmtflags showbase
- static const fmtflags showpoint
- static const fmtflags showpos
- static const fmtflags skipws
- static const openmode trunc
- static const fmtflags unitbuf
- static const fmtflags uppercase

Protected Types

- enum

Protected Member Functions

- void **_M_cache_locale** (const locale &__loc)
- void **_M_call_callbacks** (event __ev) throw ()
- void **_M_call_callbacks** (event __ev) throw ()
- void **_M_dispose_callbacks** (void) throw ()
- _Words & **_M_grow_words** (int __index, bool __iword)
- void **_M_init** () throw ()
- __ostream_type & **_M_insert** (_ValueT __v)
- void **_M_move** (ios_base &) noexcept
- void **_M_swap** (ios_base &__rhs) noexcept
- **basic_ostream** ()
- **basic_ostream** (basic_iostream< _CharT, _Traits > &)
- **basic_ostream** (basic_ostream &&__rhs)
- **basic_ostream** (const basic_ostream &)=delete
- void init (basic_streambuf< _CharT, _Traits > *__sb)
- void **move** (basic_ios &&__rhs)
- void **move** (basic_ios &__rhs)
- basic_ostream & **operator=** (basic_ostream &&__rhs)
- basic_ostream & **operator=** (const basic_ostream &)=delete
- void **set_rdbuf** (basic_streambuf< _CharT, _Traits > *__sb)
- void **swap** (basic_ios &__rhs) noexcept
- void **swap** (basic_ostream &__rhs)

Protected Attributes

- _Callback_list * **_M_callbacks**
- const __ctype_type * **_M_ctype**
- iostate **_M_exception**
- char_type **_M_fill**
- bool **_M_fill_init**
- fmtflags **_M_flags**
- locale **_M_ios_locale**
- _Words **_M_local_word** [_S_local_word_size]
- const __num_get_type * **_M_num_get**
- const __num_put_type * **_M_num_put**
- streamsize **_M_precision**
- basic_streambuf< _CharT, _Traits > * **_M_streambuf**
- iostate **_M_streambuf_state**
- basic_ostream< _CharT, _Traits > * **_M_tie**
- streamsize **_M_width**
- _Words * **_M_word**
- int **_M_word_size**
- _Words **_M_word_zero**

5.881.1 Detailed Description

template<typename _CharT, typename _Traits>
class std::basic_ostream< _CharT, _Traits >::sentry

Performs setup work for output streams.
Objects of this class are created before all of the standard inserters are run. It is responsible for *exception-safe prefix and suffix operations*.

5.881.2 Member Typedef Documentation

__num_get_type

```
typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits> > std::basic_ios< _CharT, _Traits
>::__num_get_type
```

These are non-standard types.

event_callback

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i)
```

The type of an event callback function.

Parameters

_↩ _*e*	One of the members of the event enum.
_↩ _*b*	Reference to the ios_base object.
_↩ _*i*	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several ios_base and basic_ios functions, specifically imbue(), copyfmt(), and ~ios().

iostate

`typedef _Ios_Iostate std::ios_base::iostate`
This is a bitmask type.
_Ios_Iostate is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit
- eofbit
- failbit
- goodbit

openmode

`typedef _Ios_Openmode std::ios_base::openmode`
This is a bitmask type.
_Ios_Openmode is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type openmode are:

- app
- ate
- binary
- in
- out
- trunc

seekdir

`typedef _Ios_Seekdir std::ios_base::seekdir`
This is an enumerated type.
_Ios_Seekdir is implementation-defined. Defined values of type seekdir are:

- beg
- cur, equivalent to SEEK_CUR in the C standard library.
- end, equivalent to SEEK_END in the C standard library.

5.881.3 Member Enumeration Documentation

event

`enum std::ios_base::event`
The set of events that may be passed to an event callback.
erase_event is used during ~ios() and copyfmt(). imbue_event is used during imbue(). copyfmt_event is used during copyfmt().

5.881.4 Constructor & Destructor Documentation

sentry()

```
template<typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits >::sentry::sentry (
            basic_ostream< _CharT, _Traits > & __os)  [explicit]
```

The constructor performs preparatory work.

Parameters

__os	The output stream to guard.

If the stream state is good (*__os.good()* is true), then if the stream is tied to another output stream, `is.tie()->flush()` is called to synchronize the output sequences.
If the stream state is still good, then the sentry state becomes true (*okay*).
References std::basic_ios< _CharT, _Traits >::bad(), basic_ostream(), std::ios_base::failbit, std::basic_ios< _CharT, _Traits >::good(), std::basic_ios< _CharT, _Traits >::setstate(), and std::basic_ios< _CharT, _Traits >::tie().

~sentry()

```
template<typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits >::sentry::~sentry ()  [inline]
```

Possibly flushes the stream.
If `ios_base::unitbuf` is set in `os.flags()`, and `std::uncaught_exception()` is true, the sentry destructor flushes the output stream.
References std::ios_base::badbit, std::uncaught_exception(), and std::ios_base::unitbuf.

~basic_ostream()

```
virtual std::basic_ostream< _CharT, _Traits >::~basic_ostream ()  [inline], [virtual]
```

Base destructor.
This does very little apart from providing a virtual base dtor.

5.881.5 Member Function Documentation

_M_getloc()

```
const locale & std::ios_base::_M_getloc () const  [inline]
```

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.

bad()

```
bool std::basic_ios< _CharT, _Traits >::bad () const  [inline], [nodiscard]
```

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.

basic_ostream()

```
std::basic_ostream< _CharT, _Traits >::basic_ostream (
            __streambuf_type * __sb)  [inline], [explicit]
```

Base constructor.
This ctor is almost never called by the user directly, rather from derived classes' initialization lists, which pass a pointer to their own stream buffer.
Referenced by sentry().

clear()

```
void std::basic_ios< _CharT, _Traits >::clear (
            iostate __state = goodbit)
```

[Re]sets the error state.

Parameters

__state	The new state flag(s) to set.

See std::ios_base::iostate for the possible bit values. Most users will not need to pass an argument.

copyfmt()

```
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
            const basic_ios & __rhs)
```

Copies fields of __rhs into this.

Parameters

__rhs	The source values for the copies.

Returns

Reference to this object.

All fields of __rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase_event. After copying, each (new) callback is invoked with copyfmt_event. The final step is to copy exceptions().

eof()

```
bool std::basic_ios< _CharT, _Traits >::eof () const  [inline], [nodiscard]
```

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

exceptions() [1/2]

```
iostate std::basic_ios< _CharT, _Traits >::exceptions () const  [inline], [nodiscard]
```

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.

exceptions() [2/2]

```
void std::basic_ios< _CharT, _Traits >::exceptions (
            iostate __except)  [inline]
```

Throwing exceptions on errors.

Parameters

__except	The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type std::ios_base::failure is thrown.
If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr « "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr « "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

fail()

```
bool std::basic_ios< _CharT, _Traits >::fail () const  [inline], [nodiscard]
```

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.

fill() [1/2]

```
char_type std::basic_ios< _CharT, _Traits >::fill () const  [inline], [nodiscard]
```

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.

fill() [2/2]

```
char_type std::basic_ios< _CharT, _Traits >::fill (
            char_type __ch)  [inline]
```

Sets a new *empty* character.

Parameters

__ch	The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

flags() [1/2]

`fmtflags std::ios_base::flags () const [inline], [nodiscard]`
Access to format flags.

Returns

The format control flags for both input and output.

flags() [2/2]

`fmtflags std::ios_base::flags (`
`fmtflags __fmtfl) [inline]`
Setting new format flags all at once.

Parameters

__fmtfl	The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *__fmtfl*.

flush()

`basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::flush ()`
Synchronizing the stream buffer.

Returns

*this

If `rdbuf()` is a null pointer, changes nothing.
Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit.

getloc()

`locale std::ios_base::getloc () const [inline], [nodiscard]`
Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.

good()

```
bool std::basic_ios< _CharT, _Traits >::good () const  [inline], [nodiscard]
```

Fast error checking.

Returns

True if no error flags are set.

A wrapper around rdstate.

imbue()

```
locale std::basic_ios< _CharT, _Traits >::imbue (
            const locale & __loc)
```

Moves to a new locale.

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.
Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html

init()

```
void std::basic_ios< _CharT, _Traits >::init (
            basic_streambuf< _CharT, _Traits > * __sb)  [protected]
```

All setup is performed here.
This is called from the public constructor. It is not virtual and cannot be redefined.

iword()

```
long & std::ios_base::iword (
            int __ix)  [inline]
```

Access to integer array.

Parameters

__ix	Index into the array.

Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

narrow()

```
char std::basic_ios< _CharT, _Traits >::narrow (
            char_type __c,
            char __dfault) const  [inline]
```

Squeezes characters.

Parameters

__c	The character to narrow.
__dfault	The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.
Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c,dfault)
```

Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html

operator bool() [1/2]

```
std::basic_ios< _CharT, _Traits >::operator bool () const  [inline], [explicit], [nodiscard]
```

The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

operator bool() [2/2]

```
template<typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits >::sentry::operator bool () const  [inline], [explicit]
```

Quick status checking.

Returns

The sentry state.

For ease of use, sentries may be converted to booleans. The return value is that of the sentry state (true == okay).

operator"!()

```
bool std::basic_ios< _CharT, _Traits >::operator!  () const  [inline], [nodiscard]
```

The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

operator<<() [1/17]

```
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            __ios_type &(* __pf )(__ios_type &))  [inline]
```

Interface for manipulators.
Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

operator<<() [2/17]

```
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            __ostream_type &(* __pf )(__ostream_type &))  [inline]
```

Interface for manipulators.
Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

operator<<() [3/17]

```
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
            __streambuf_type * __sb)
```

Extracting from another streambuf.

Parameters

__sb	A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.
If `__sb` is NULL, the stream will set failbit in its error state.
Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets failbit in the error state

If the function inserts no characters, failbit is set.

operator<<() [4/17]

```
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            bool __n)  [inline]
```

Integer arithmetic inserters.

Parameters

__n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [5/17]

```
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            const void * __p)  [inline]
```

Pointer arithmetic inserters.

Parameters

_↩ _p	A variable of pointer type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [6/17]

```
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            double __f)  [inline]
```

Floating point arithmetic inserters.

Parameters

↩ _↩ ↩ _↩ f	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [7/17]

```
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            float __f)  [inline]
```

Floating point arithmetic inserters.

Parameters

↩ _↩ ↩ _↩ f	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [8/17]

```
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
            int __n)
```

Integer arithmetic inserters.

Parameters

_↩ _*n*	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [9/17]

```
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            ios_base &(* __pf )(ios_base &))  [inline]
```

Interface for manipulators.
Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

operator<<() [10/17]

```
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            long __n)  [inline]
```

Integer arithmetic inserters.

Parameters

_↩ _*n*	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [11/17]

```
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            long double __f)  [inline]
```

Floating point arithmetic inserters.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [12/17]

```
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            long long __n)  [inline]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [13/17]

```
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
            short __n)
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [14/17]

```
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned int __n)  [inline]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [15/17]

```
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned long __n)  [inline]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [16/17]

```
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned long long __n)  [inline]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [17/17]

```
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned short __n)  [inline]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

precision() [1/2]

```
streamsize std::ios_base::precision () const  [inline], [nodiscard]
```

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.

precision() [2/2]

```
streamsize std::ios_base::precision (
            streamsize __prec)  [inline]
```

Changing flags.

Parameters

__prec	The new precision value.

Returns

The previous value of precision().

put()

```
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (
            char_type __c)
```

Simple insertion.

Parameters

_↩ _c	The character to insert.

Returns

*this

Tries to insert __c.

Note

This function is not overloaded on signed char and unsigned char.

pword()

```
void *& std::ios_base::pword (
            int __ix)  [inline]
```

Access to void pointer array.

Parameters

_↩ _ix	Index into the array.

Returns

A reference to a void* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

rdbuf() [1/2]

```
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf () const  [inline],
[nodiscard]
```

Accessing the underlying buffer.

Returns

The current stream buffer.

This does not change the state of the stream.

rdbuf() [2/2]

```
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
            basic_streambuf< _CharT, _Traits > * __sb)
```

Changing the underlying buffer.

Parameters

__sb	The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.
Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream     foo;         // or some other derived type
std::streambuf*  p = .....;

foo.ios::rdbuf(p);            // ios == basic_ios<char>
```

rdstate()

```
iostate std::basic_ios< _CharT, _Traits >::rdstate () const  [inline], [nodiscard]
```

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See std::ios_base::iostate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().

register_callback()

```
void std::ios_base::register_callback (
            event_callback __fn,
            int __index)
```

Add the callback __fn with parameter __index.

Parameters

__fn	The function to add.
__index	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

seekp() [1/2]

```
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
            off_type __off,
            ios_base::seekdir __dir)
```

Changing the current write position.

Parameters

__off	A file offset object.
__dir	The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off,dir)`. If that function fails, sets failbit.

seekp() [2/2]

```
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
            pos_type __pos)
```

Changing the current write position.

Parameters

__pos	A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.

setf() [1/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl)  [inline]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.

setf() [2/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl,
            fmtflags __mask)  [inline]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.
__mask	The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

setstate()

```
void std::basic_ios< _CharT, _Traits >::setstate (
            iostate __state)  [inline]
```

Sets additional flags in the error state.

Parameters

__state	The additional state flag(s) to set.

See std::ios_base::iostate for the possible bit values.

sync_with_stdio()

```
static bool std::ios_base::sync_with_stdio (
            bool __sync = true)  [static]
```

Interaction with the standard C I/O objects.

Parameters

__sync	Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary

tellp()

```
basic_ostream< _CharT, _Traits >::pos_type std::basic_ostream< _CharT, _Traits >::tellp ()
```

Getting the current write position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type`(-1) to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0,cur,out)`.

tie() [1/2]

```
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie () const  [inline],
[nodiscard]
```

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.

tie() [2/2]

```
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie (
            basic_ostream< _CharT, _Traits > * __tiestr)  [inline]
```

Ties this stream to an output stream.

Parameters

__tiestr	The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

unsetf()

```
void std::ios_base::unsetf (
            fmtflags __mask)  [inline]
```

Clearing format flags.

Parameters

__mask	The flags to unset.

This function clears *__mask* in the format flags.

widen()

```
char_type std::basic_ios< _CharT, _Traits >::widen (
            char __c) const  [inline]
```

Widens characters.

Parameters

_c	The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.
Returns the result of
`std::use_facet<ctype<char_type> >(getloc()).widen(c)`
Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html

width() [1/2]

```
streamsize std::ios_base::width () const  [inline], [nodiscard]
```

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.

width() [2/2]

```
streamsize std::ios_base::width (
            streamsize __wide)  [inline]
```

Changing flags.

Parameters

__wide	The new width value.

Returns

The previous value of width().

write()

```
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::write (
            const char_type * __s,
            streamsize __n)
```

Character string insertion.

Parameters

_↩ _s	The array to insert.
_↩ _n	Maximum number of characters to insert.

Returns

*this

Characters are copied from __s and inserted into the stream until one of the following happens:

- __n characters are inserted
- inserting into the output sequence fails (in this case, badbit will be set in the stream's error state)

Note

This function is not overloaded on signed char and unsigned char.

xalloc()

```
static int std::ios_base::xalloc () throw ( )   [static]
```

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the iword and pword functions. The expectation is that an application calls xalloc in order to obtain an index in the iword and pword arrays that can be used without fear of conflict.
The implementation maintains a static variable that is incremented and returned on each invocation. xalloc is guaranteed to return an index that is safe to use in the iword and pword arrays.

5.881.6 Member Data Documentation

adjustfield

```
const fmtflags std::ios_base::adjustfield  [static]
```
A mask of left|right|internal. Useful for the 2-arg form of `setf`.

app

```
const openmode std::ios_base::app  [static]
```
Seek to end before each write.

ate

```
const openmode std::ios_base::ate  [static]
```
Open and seek to end immediately after opening.

badbit

```
const iostate std::ios_base::badbit  [static]
```
Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).

basefield

```
const fmtflags std::ios_base::basefield  [static]
```
A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.

beg

```
const seekdir std::ios_base::beg  [static]
```
Request a seek relative to the beginning of the stream.

binary

```
const openmode std::ios_base::binary  [static]
```
Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary.

boolalpha

```
const fmtflags std::ios_base::boolalpha  [static]
```
Insert/extract `bool` in alphabetic rather than numeric format.

cur

```
const seekdir std::ios_base::cur  [static]
```
Request a seek relative to the current position within the sequence.

dec

```
const fmtflags std::ios_base::dec  [static]
```
Converts integer input or generates integer output in decimal base.

end

```
const seekdir std::ios_base::end [static]
```
Request a seek relative to the current end of the sequence.

eofbit

```
const iostate std::ios_base::eofbit [static]
```
Indicates that an input operation reached the end of an input sequence.

failbit

```
const iostate std::ios_base::failbit [static]
```
Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

fixed

```
const fmtflags std::ios_base::fixed [static]
```
Generate floating-point output in fixed-point notation.

floatfield

```
const fmtflags std::ios_base::floatfield [static]
```
A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

fmtflags

```
class __attribute ((__abi_tag__ ("cxx11"))) failure typedef _Ios_Fmtflags std::ios_base::fmtflags
```
These are thrown to indicate problems with io.
27.4.2.1.1 Class ios_base::failure
This is a bitmask type.
_Ios_Fmtflags is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type fmtflags are:

- boolalpha
- dec
- fixed
- hex
- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws

- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

goodbit

`const iostate std::ios_base::goodbit [static]`
Indicates all is well.

hex

`const fmtflags std::ios_base::hex [static]`
Converts integer input or generates integer output in hexadecimal base.

in

`const openmode std::ios_base::in [static]`
Open for input. Default for `ifstream` and fstream.

internal

`const fmtflags std::ios_base::internal [static]`
Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

left

`const fmtflags std::ios_base::left [static]`
Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.)

oct

`const fmtflags std::ios_base::oct [static]`
Converts integer input or generates integer output in octal base.

out

`const openmode std::ios_base::out [static]`
Open for output. Default for `ofstream` and fstream.

right

`const fmtflags std::ios_base::right [static]`
Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.)

scientific

`const fmtflags std::ios_base::scientific [static]`
Generates floating-point output in scientific notation.

showbase

`const fmtflags std::ios_base::showbase [static]`
Generates a prefix indicating the numeric base of generated integer output.

showpoint

`const fmtflags std::ios_base::showpoint [static]`
Generates a decimal-point character unconditionally in generated floating-point output.

showpos

`const fmtflags std::ios_base::showpos [static]`
Generates a + sign in non-negative generated numeric output.

skipws

`const fmtflags std::ios_base::skipws [static]`
Skips leading white space before certain input operations.

trunc

`const openmode std::ios_base::trunc [static]`
Truncate an existing stream when opening. Default for `ofstream`.

unitbuf

`const fmtflags std::ios_base::unitbuf [static]`
Flushes output after each output operation.

uppercase

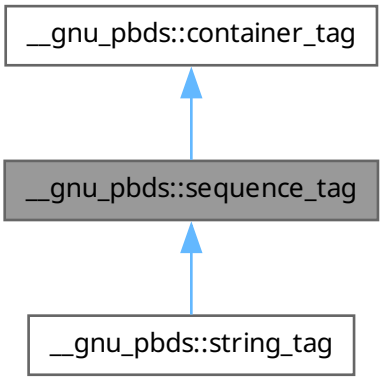
`const fmtflags std::ios_base::uppercase [static]`
Replaces certain lowercase letters with their uppercase equivalents in generated output.
The documentation for this class was generated from the following files:

- ostream.h
- ostream.tcc

5.882 __gnu_pbds::sequence_tag Struct Reference

`#include <tag_and_trait.hpp>`

Inheritance diagram for __gnu_pbds::sequence_tag:



5.882.1 Detailed Description

Basic sequence.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

5.883 __gnu_parallel::sequential_tag Struct Reference

`#include <tags.h>`

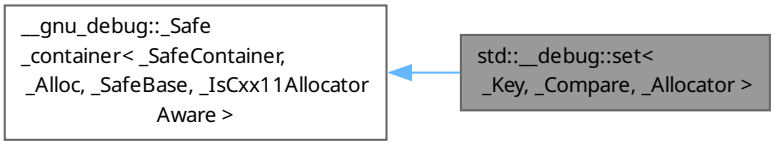
5.883.1 Detailed Description

Forces sequential execution at compile time.
The documentation for this struct was generated from the following file:

- tags.h

5.884 std::__debug::set< _Key, _Compare, _Allocator > Class Template Reference

`#include <set.h>`
Inheritance diagram for std::__debug::set< _Key, _Compare, _Allocator >:



Public Types

- typedef _Allocator **allocator_type**
- typedef __gnu_debug::_Safe_iterator< _Base_const_iterator, set > **const_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**
- typedef std::reverse_iterator< const_iterator > **const_reverse_iterator**
- typedef _Base::difference_type **difference_type**
- using **insert_return_type**
- typedef __gnu_debug::_Safe_iterator< _Base_iterator, set > **iterator**
- typedef _Compare **key_compare**
- typedef _Key **key_type**
- using **node_type**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef std::reverse_iterator< iterator > **reverse_iterator**
- typedef _Base::size_type **size_type**
- typedef _Compare **value_compare**
- typedef _Key **value_type**

Public Member Functions

- **set** (_Base_ref __x)
- template<typename _InputIterator>
  set (_InputIterator __first, _InputIterator __last, const _Compare &__comp=_Compare(), const _Allocator &__↩ a=_Allocator())
- template<typename _InputIterator>
  set (_InputIterator __first, _InputIterator __last, const allocator_type &__a)
- **set** (const _Compare &__comp, const _Allocator &__a=_Allocator())
- **set** (const allocator_type &__a)
- **set** (const set &)=default
- **set** (const set &__x, const __type_identity_t< allocator_type > &__a)
- **set** (initializer_list< value_type > __l, const _Compare &__comp=_Compare(), const allocator_type &__↩ a=allocator_type())
- **set** (initializer_list< value_type > __l, const allocator_type &__a)
- **set** (set &&)=default
- **set** (set &&__x, const __type_identity_t< allocator_type > &__a) noexcept(noexcept(_Base(std::move(__x), _↩ _a)))
- **__attribute** ((__abi_tag__("cxx11"))) iterator erase(const_iterator __first
- **__attribute** ((__abi_tag__("cxx11"))) iterator erase(const_iterator __position)
- const _Base & **_M_base** () const noexcept
- _Base & **_M_base** () noexcept
- const_iterator **begin** () const noexcept
- iterator **begin** () noexcept
- const_iterator **cbegin** () const noexcept
- const_iterator **cend** () const noexcept
- void **clear** () noexcept
- const_reverse_iterator **crbegin** () const noexcept
- const_reverse_iterator **crend** () const noexcept
- template<typename... _Args>
  std::pair< iterator, bool > **emplace** (_Args &&... __args)

- template<typename... _Args>
  iterator **emplace_hint** (const_iterator __pos, _Args &&... __args)
- const_iterator **end** () const noexcept
- iterator **end** () noexcept
- template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>
  std::pair< iterator, iterator > **equal_range** (const _Kt &__x)
- template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>
  std::pair< const_iterator, const_iterator > **equal_range** (const _Kt &__x) const
- std::pair< iterator, iterator > **equal_range** (const key_type &__x)
- std::pair< const_iterator, const_iterator > **equal_range** (const key_type &__x) const
- _Base_iterator **erase** (_Base_const_iterator __position)
- size_type **erase** (const key_type &__x)
- node_type **extract** (const key_type &__key)
- node_type **extract** (const_iterator __position)
- template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>
  iterator **find** (const _Kt &__x)
- template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>
  const_iterator **find** (const _Kt &__x) const
- iterator **find** (const key_type &__x)
- const_iterator **find** (const key_type &__x) const
- **for** (_Base_const_iterator __victim=__first.base();__victim !=__last.base();++__victim)
- template<typename _InputIterator>
  void **insert** (_InputIterator __first, _InputIterator __last)
- std::pair< iterator, bool > **insert** (const value_type &__x)
- iterator **insert** (const_iterator __hint, node_type &&__nh)
- iterator **insert** (const_iterator __position, const value_type &__x)
- iterator **insert** (const_iterator __position, value_type &&__x)
- void **insert** (initializer_list< value_type > __l)
- insert_return_type **insert** (node_type &&__nh)
- std::pair< iterator, bool > **insert** (value_type &&__x)
- template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>
  iterator **lower_bound** (const _Kt &__x)
- template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>
  const_iterator **lower_bound** (const _Kt &__x) const
- iterator **lower_bound** (const key_type &__x)
- const_iterator **lower_bound** (const key_type &__x) const
- set & **operator=** (const set &)=default
- set & **operator=** (initializer_list< value_type > __l)
- set & **operator=** (set &&)=default
- const_reverse_iterator **rbegin** () const noexcept
- reverse_iterator **rbegin** () noexcept
- const_reverse_iterator **rend** () const noexcept
- reverse_iterator **rend** () noexcept
- void **swap** (set &__x) noexcept(/*conditional */)
- template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>
  iterator **upper_bound** (const _Kt &__x)
- template<typename _Kt, typename _Req = typename __has_is_transparent<_Compare, _Kt>::type>
  const_iterator **upper_bound** (const _Kt &__x) const
- iterator **upper_bound** (const key_type &__x)
- const_iterator **upper_bound** (const key_type &__x) const
- **while** (false)

Public Attributes

- const_iterator **__last**
- **do**
- **return**

Protected Member Functions

- constexpr void **_M_swap** (const _Safe_container &__x) const noexcept

Friends

- template<typename _ItT, typename _SeqT, typename _CatT>
  class **::__gnu_debug::_Safe_iterator**

5.884.1 Detailed Description

template<typename _Key, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<_↩ Key>>
class std::__debug::set< _Key, _Compare, _Allocator >

Class std::set with safety/checking/debug instrumentation.
The documentation for this class was generated from the following file:

- set.h

5.885 std::set< _Key, _Compare, _Alloc > Class Template Reference

`#include <set>`

Public Types

- typedef _Key key_type
- typedef _Key value_type
- typedef _Compare key_compare
- typedef _Compare value_compare
- typedef _Alloc allocator_type

- typedef _Alloc_traits::pointer pointer
- typedef _Alloc_traits::const_pointer const_pointer
- typedef _Alloc_traits::reference reference
- typedef _Alloc_traits::const_reference const_reference
- typedef _Rep_type::const_iterator iterator
- typedef _Rep_type::const_iterator const_iterator
- typedef _Rep_type::const_reverse_iterator reverse_iterator
- typedef _Rep_type::const_reverse_iterator const_reverse_iterator
- typedef _Rep_type::size_type size_type
- typedef _Rep_type::difference_type difference_type

Public Member Functions

- set ()=default
- template<typename _InputIterator>
  set (_InputIterator __first, _InputIterator __last)
- template<typename _InputIterator>
  set (_InputIterator __first, _InputIterator __last, const _Compare &__comp, const allocator_type &__a=allocator_type())
- template<typename _InputIterator>
  set (_InputIterator __first, _InputIterator __last, const allocator_type &__a)
- set (const _Compare &__comp, const allocator_type &__a=allocator_type())
- set (const allocator_type &__a)
- set (const set &)=default
- set (const set &__x, const __type_identity_t< allocator_type > &__a)
- set (initializer_list< value_type > __l, const _Compare &__comp=_Compare(), const allocator_type &__a=allocator_type())
- set (initializer_list< value_type > __l, const allocator_type &__a)
- set (set &&)=default
- set (set &&__x, const __type_identity_t< allocator_type > &__a) noexcept(is_nothrow_copy_constructible< _Compare >::value &&_Alloc_traits::_S_always_equal())
- ~set ()=default
- iterator begin () const noexcept
- iterator cbegin () const noexcept
- iterator cend () const noexcept
- void clear () noexcept
- reverse_iterator crbegin () const noexcept
- reverse_iterator crend () const noexcept
- template<typename... _Args>
  std::pair< iterator, bool > emplace (_Args &&... __args)
- template<typename... _Args>
  iterator emplace_hint (const_iterator __pos, _Args &&... __args)
- bool empty () const noexcept
- iterator end () const noexcept
- size_type erase (const key_type &__x)
- _GLIBCXX_ABI_TAG_CXX11 iterator erase (const_iterator __first, const_iterator __last)
- _GLIBCXX_ABI_TAG_CXX11 iterator erase (const_iterator __position)
- allocator_type get_allocator () const noexcept
- template<typename _InputIterator>
  void insert (_InputIterator __first, _InputIterator __last)
- std::pair< iterator, bool > insert (const value_type &__x)
- iterator insert (const_iterator __position, const value_type &__x)
- iterator **insert** (const_iterator __position, value_type &&__x)
- void insert (initializer_list< value_type > __l)
- std::pair< iterator, bool > **insert** (value_type &&__x)
- key_compare key_comp () const
- size_type max_size () const noexcept
- set & operator= (const set &)=default
- set & operator= (initializer_list< value_type > __l)
- set & operator= (set &&)=default
- reverse_iterator rbegin () const noexcept
- reverse_iterator rend () const noexcept
- size_type size () const noexcept

- void swap (set &__x) noexcept(/*conditional */)
- value_compare value_comp () const

- size_type count (const key_type &__x) const
- template<typename _Kt>
  auto count (const _Kt &__x) const -> decltype(_M_t._M_count_tr(__x))

- bool contains (const key_type &__x) const
- template<typename _Kt>
  auto contains (const _Kt &__x) const -> decltype(_M_t._M_find_tr(__x), void(), true)

- iterator find (const key_type &__x)
- const_iterator find (const key_type &__x) const
- template<typename _Kt>
  auto find (const _Kt &__x) -> decltype(iterator{_M_t._M_find_tr(__x)})
- template<typename _Kt>
  auto find (const _Kt &__x) const -> decltype(const_iterator{_M_t._M_find_tr(__x)})

- iterator lower_bound (const key_type &__x)
- const_iterator lower_bound (const key_type &__x) const
- template<typename _Kt>
  auto lower_bound (const _Kt &__x) -> decltype(iterator(_M_t._M_lower_bound_tr(__x)))
- template<typename _Kt>
  auto lower_bound (const _Kt &__x) const -> decltype(const_iterator(_M_t._M_lower_bound_tr(__x)))

- iterator upper_bound (const key_type &__x)
- const_iterator upper_bound (const key_type &__x) const
- template<typename _Kt>
  auto upper_bound (const _Kt &__x) -> decltype(iterator(_M_t._M_upper_bound_tr(__x)))
- template<typename _Kt>
  auto upper_bound (const _Kt &__x) const -> decltype(const_iterator(_M_t._M_upper_bound_tr(__x)))

- std::pair< iterator, iterator > equal_range (const key_type &__x)
- std::pair< const_iterator, const_iterator > equal_range (const key_type &__x) const
- template<typename _Kt>
  auto equal_range (const _Kt &__x) -> decltype(pair< iterator, iterator >(_M_t._M_equal_range_tr(__x)))
- template<typename _Kt>
  auto equal_range (const _Kt &__x) const -> decltype(pair< iterator, iterator >(_M_t._M_equal_range_tr(__x)))

Friends

- template<typename _K1, typename _C1, typename _A1>
  bool **operator<** (const set< _K1, _C1, _A1 > &, const set< _K1, _C1, _A1 > &)
- template<typename _K1, typename _C1, typename _A1>
  bool **operator==** (const set< _K1, _C1, _A1 > &, const set< _K1, _C1, _A1 > &)

5.885.1 Detailed Description

template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>> class std::set< _Key, _Compare, _Alloc >

A standard container made up of unique keys, which can be retrieved in logarithmic time.

Since

C++98

Template Parameters

_Key	Type of key objects.
_Compare	Comparison function object type, defaults to less<_Key>.
_Alloc	Allocator type, defaults to allocator<_Key>.

Meets the requirements of a container, a reversible container, and an associative container (using unique keys).
Sets support bidirectional iterators.
The private tree data is declared exactly the same way for set and multiset; the distinction is made entirely in how the tree functions are called (*_unique versus *_equal, same as the standard).

5.885.2 Member Typedef Documentation

allocator_type

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
typedef _Alloc std::set< _Key, _Compare, _Alloc >::allocator_type
```

Public typedefs.

const_iterator

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
typedef _Rep_type::const_iterator std::set< _Key, _Compare, _Alloc >::const_iterator
```

Iterator-related typedefs.

const_pointer

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
typedef _Alloc_traits::const_pointer std::set< _Key, _Compare, _Alloc >::const_pointer
```

Iterator-related typedefs.

const_reference

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
typedef _Alloc_traits::const_reference std::set< _Key, _Compare, _Alloc >::const_reference
```

Iterator-related typedefs.

const_reverse_iterator

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
typedef _Rep_type::const_reverse_iterator std::set< _Key, _Compare, _Alloc >::const_reverse_↩
iterator
```

Iterator-related typedefs.

difference_type

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
typedef _Rep_type::difference_type std::set< _Key, _Compare, _Alloc >::difference_type
```

Iterator-related typedefs.

iterator

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
typedef _Rep_type::const_iterator std::set< _Key, _Compare, _Alloc >::iterator
```

Iterator-related typedefs.

key_compare

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
typedef _Compare std::set< _Key, _Compare, _Alloc >::key_compare
```

Public typedefs.

key_type

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
typedef _Key std::set< _Key, _Compare, _Alloc >::key_type
```

Public typedefs.

pointer

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
typedef _Alloc_traits::pointer std::set< _Key, _Compare, _Alloc >::pointer
```

Iterator-related typedefs.

reference

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
typedef _Alloc_traits::reference std::set< _Key, _Compare, _Alloc >::reference
```

Iterator-related typedefs.

reverse_iterator

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
typedef _Rep_type::const_reverse_iterator std::set< _Key, _Compare, _Alloc >::reverse_iterator
```

Iterator-related typedefs.

size_type

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
typedef _Rep_type::size_type std::set< _Key, _Compare, _Alloc >::size_type
```

Iterator-related typedefs.

value_compare

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
typedef _Compare std::set< _Key, _Compare, _Alloc >::value_compare
```

Public typedefs.

value_type

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
typedef _Key std::set< _Key, _Compare, _Alloc >::value_type
```

Public typedefs.

5.885.3 Constructor & Destructor Documentation

set() [1/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::set< _Key, _Compare, _Alloc >::set ()  [default]
```

Default constructor creates no elements.

set() [2/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::set< _Key, _Compare, _Alloc >::set (
            const _Compare & __comp,
            const allocator_type & __a = allocator_type())  [inline], [explicit]
```

Creates a set with no elements.

Parameters

__comp	Comparator to use.
__a	An allocator object.

set() [3/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
template<typename _InputIterator>
std::set< _Key, _Compare, _Alloc >::set (
            _InputIterator __first,
            _InputIterator __last)  [inline]
```

Builds a set from a range.

Parameters

__first	An input iterator.
__last	An input iterator.

Create a set consisting of copies of the elements from [__first,__last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(__first,__last)).

set() [4/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
template<typename _InputIterator>
std::set< _Key, _Compare, _Alloc >::set (
            _InputIterator __first,
            _InputIterator __last,
            const _Compare & __comp,
            const allocator_type & __a = allocator_type())  [inline]
```

Builds a set from a range.

Parameters

__first	An input iterator.
__last	An input iterator.
__comp	A comparison functor.
__a	An allocator object.

Create a set consisting of copies of the elements from [__first,__last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(__first,__last)).

set() [5/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::set< _Key, _Compare, _Alloc >::set (
            const set< _Key, _Compare, _Alloc > & )  [default]
```

Set copy constructor.
Whether the allocator is copied depends on the allocator traits.

set() [6/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::set< _Key, _Compare, _Alloc >::set (
            set< _Key, _Compare, _Alloc > && )  [default]
```

Set move constructor
The newly-created set contains the exact contents of the moved instance. The moved instance is a valid, but unspecified, set.

set() [7/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
```

```
std::set< _Key, _Compare, _Alloc >::set (
            initializer_list< value_type > __l,
            const _Compare & __comp = _Compare(),
            const allocator_type & __a = allocator_type())  [inline]
```

Builds a set from an initializer_list.

Parameters

__l	An initializer_list.
__comp	A comparison functor.
__a	An allocator object.

Create a set consisting of copies of the elements in the list. This is linear in N if the list is already sorted, and NlogN otherwise (where N is *__l.size()*).

set() [8/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::set< _Key, _Compare, _Alloc >::set (
            const allocator_type & __a)  [inline], [explicit]
```

Allocator-extended default constructor.

set() [9/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::set< _Key, _Compare, _Alloc >::set (
            const set< _Key, _Compare, _Alloc > & __x,
            const __type_identity_t< allocator_type > & __a)  [inline]
```

Allocator-extended copy constructor.

set() [10/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::set< _Key, _Compare, _Alloc >::set (
            set< _Key, _Compare, _Alloc > && __x,
            const __type_identity_t< allocator_type > & __a)  [inline], [noexcept]
```

Allocator-extended move constructor.

set() [11/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::set< _Key, _Compare, _Alloc >::set (
            initializer_list< value_type > __l,
            const allocator_type & __a)  [inline]
```

Allocator-extended initialier-list constructor.

set() [12/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
```

```
template<typename _InputIterator>
std::set< _Key, _Compare, _Alloc >::set (
            _InputIterator __first,
            _InputIterator __last,
            const allocator_type & __a)  [inline]
```

Allocator-extended range constructor.

~set()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::set< _Key, _Compare, _Alloc >::~set ()  [default]
```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

5.885.4 Member Function Documentation

begin()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
iterator std::set< _Key, _Compare, _Alloc >::begin () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the set. Iteration is done in ascending order according to the keys.

cbegin()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
iterator std::set< _Key, _Compare, _Alloc >::cbegin () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the set. Iteration is done in ascending order according to the keys.

cend()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
iterator std::set< _Key, _Compare, _Alloc >::cend () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the set. Iteration is done in ascending order according to the keys.

clear()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
void std::set< _Key, _Compare, _Alloc >::clear ()  [inline], [noexcept]
```

Erases all elements in a set. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

contains() [1/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
template<typename _Kt>
auto std::set< _Key, _Compare, _Alloc >::contains (
            const _Kt & __x) const -> decltype(_M_t._M_find_tr(__x), void(), true)   [inline]
```

Finds whether an element with the given key exists.

Parameters

_↩ _x	Key of elements to be located.

Returns

True if there is an element with the specified key.

contains() [2/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
bool std::set< _Key, _Compare, _Alloc >::contains (
            const key_type & __x) const  [inline]
```

Finds whether an element with the given key exists.

Parameters

_↩ _x	Key of elements to be located.

Returns

True if there is an element with the specified key.

count() [1/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
template<typename _Kt>
auto std::set< _Key, _Compare, _Alloc >::count (
            const _Kt & __x) const -> decltype(_M_t._M_count_tr(__x))   [inline]
```

Finds the number of elements.

Parameters

_↩ _x	Element to located.

Returns

Number of elements with specified key.

This function only makes sense for multisets; for set the result will either be 0 (not present) or 1 (present).

count() [2/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
size_type std::set< _Key, _Compare, _Alloc >::count (
            const key_type & __x) const  [inline]
```

Finds the number of elements.

Parameters

_↩ _x	Element to located.

Returns

Number of elements with specified key.

This function only makes sense for multisets; for set the result will either be 0 (not present) or 1 (present).

crbegin()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
reverse_iterator std::set< _Key, _Compare, _Alloc >::crbegin () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the last element in the set. Iteration is done in descending order according to the keys.

crend()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
reverse_iterator std::set< _Key, _Compare, _Alloc >::crend () const  [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the set. Iteration is done in descending order according to the keys.

emplace()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
template<typename...  _Args>
std::pair< iterator, bool > std::set< _Key, _Compare, _Alloc >::emplace (
            _Args &&...  __args)  [inline]
```

Attempts to build and insert an element into the set.

Parameters

__args	Arguments used to generate an element.

Returns

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to build and insert an element into the set. A set relies on unique keys and thus an element is only inserted if it is not already present in the set.
Insertion requires logarithmic time.

emplace_hint()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
template<typename...  _Args>
iterator std::set< _Key, _Compare, _Alloc >::emplace_hint (
```

```
            const_iterator __pos,
            _Args &&... __args)  [inline]
```

Attempts to insert an element into the set.

Parameters

__pos	An iterator that serves as a hint as to where the element should be inserted.
__args	Arguments used to generate the element to be inserted.

Returns

An iterator that points to the element with key equivalent to the one generated from *__args* (may or may not be the element itself).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument emplace() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
For more on *hinting*, see: https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints
Insertion requires logarithmic time (if the hint is not taken).

empty()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
bool std::set< _Key, _Compare, _Alloc >::empty () const  [inline], [nodiscard], [noexcept]
```

Returns true if the set is empty.

end()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
iterator std::set< _Key, _Compare, _Alloc >::end () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the set. Iteration is done in ascending order according to the keys.

equal_range() [1/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _Kt>
auto std::set< _Key, _Compare, _Alloc >::equal_range (
            const _Kt & __x) -> decltype(pair<iterator, iterator>(_M_t._M_equal_range_tr(__x)))
[inline]
```

Finds a subsequence matching given key.

Parameters

_x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
               c.upper_bound(val))
```

(but is faster than making the calls separately).
This function probably only makes sense for multisets.

equal_range() [2/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
template<typename _Kt>
auto std::set< _Key, _Compare, _Alloc >::equal_range (
             const _Kt & __x) const -> decltype(pair<iterator, iterator>(_M_t._M_equal_range_↩
tr(__x)))   [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
               c.upper_bound(val))
```

(but is faster than making the calls separately).
This function probably only makes sense for multisets.

equal_range() [3/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::pair< iterator, iterator > std::set< _Key, _Compare, _Alloc >::equal_range (
             const key_type & __x)  [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
               c.upper_bound(val))
```

(but is faster than making the calls separately).
This function probably only makes sense for multisets.

equal_range() [4/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::pair< const_iterator, const_iterator > std::set< _Key, _Compare, _Alloc >::equal_range (
            const key_type & __x) const  [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
               c.upper_bound(val))
```

(but is faster than making the calls separately).
This function probably only makes sense for multisets.

erase() [1/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
size_type std::set< _Key, _Compare, _Alloc >::erase (
            const key_type & __x)  [inline]
```

Erases elements according to the provided key.

Parameters

_↩ _x	Key of element to be erased.

Returns

The number of elements erased.

This function erases all the elements located by the given key from a set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [2/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
_GLIBCXX_ABI_TAG_CXX11 iterator std::set< _Key, _Compare, _Alloc >::erase (
            const_iterator __first,
            const_iterator __last)  [inline]
```

Erases a [__first,__last) range of elements from a set.

Parameters

__first	Iterator pointing to the start of the range to be erased.

__last	Iterator pointing to the end of the range to be erased.

Returns

The iterator *__last*.

This function erases a sequence of elements from a set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [3/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
_GLIBCXX_ABI_TAG_CXX11 iterator std::set< _Key, _Compare, _Alloc >::erase (
            const_iterator __position)  [inline]
```

Erases an element from a set.

Parameters

__position	An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *__position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from a set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

find() [1/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
template<typename _Kt>
auto std::set< _Key, _Compare, _Alloc >::find (
            const _Kt & __x) -> decltype(iterator{_M_t._M_find_tr(__x)})   [inline]
```

Tries to locate an element in a set.

Parameters

_↩ _x	Element to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

find() [2/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
template<typename _Kt>
auto std::set< _Key, _Compare, _Alloc >::find (
            const _Kt & __x) const -> decltype(const_iterator{_M_t._M_find_tr(__x)})   [inline]
```

Tries to locate an element in a set.

Parameters

_↩ _x	Element to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

find() [3/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
iterator std::set< _Key, _Compare, _Alloc >::find (
            const key_type & __x)  [inline]
```

Tries to locate an element in a set.

Parameters

_↩ _x	Element to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

find() [4/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
const_iterator std::set< _Key, _Compare, _Alloc >::find (
            const key_type & __x) const  [inline]
```

Tries to locate an element in a set.

Parameters

_↩ _x	Element to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

get_allocator()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
allocator_type std::set< _Key, _Compare, _Alloc >::get_allocator () const  [inline], [noexcept]
```

Returns the allocator object with which the set was constructed.

insert() [1/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
template<typename _InputIterator>
void std::set< _Key, _Compare, _Alloc >::insert (
            _InputIterator __first,
            _InputIterator __last)  [inline]
```

A template function that attempts to insert a range of elements.

Parameters

__first	Iterator pointing to the start of the range to be inserted.
__last	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

insert() [2/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::pair< iterator, bool > std::set< _Key, _Compare, _Alloc >::insert (
            const value_type & __x)  [inline]
```

Attempts to insert an element into the set.

Parameters

_↩ _x	Element to be inserted.

Returns

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to insert an element into the set. A set relies on unique keys and thus an element is only inserted if it is not already present in the set.
Insertion requires logarithmic time.
Referenced by std::set< _Key, _Cmp, polymorphic_allocator< _Key > >::insert().

insert() [3/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
iterator std::set< _Key, _Compare, _Alloc >::insert (
            const_iterator __position,
            const value_type & __x)  [inline]
```

Attempts to insert an element into the set.

Parameters

__position	An iterator that serves as a hint as to where the element should be inserted.
__x	Element to be inserted.

Returns

An iterator that points to the element with key of *__x* (may or may not be the element passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument insert() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
For more on *hinting*, see: https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints
Insertion requires logarithmic time (if the hint is not taken).

insert() [4/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
void std::set< _Key, _Compare, _Alloc >::insert (
            initializer_list< value_type > __l)  [inline]
```

Attempts to insert a list of elements into the set.

Parameters

__l	A std::initializer_list<value_type> of elements to be inserted.

Complexity similar to that of the range constructor.

key_comp()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
key_compare std::set< _Key, _Compare, _Alloc >::key_comp () const  [inline]
```

Returns the comparison object with which the set was constructed.

lower_bound() [1/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
template<typename _Kt>
auto std::set< _Key, _Compare, _Alloc >::lower_bound (
            const _Kt & __x) -> decltype(iterator(_M_t._M_lower_bound_tr(__x)))   [inline]
```

Finds the beginning of a subsequence matching given key.

Parameters

__x	Key to be located.

Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

lower_bound() [2/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
template<typename _Kt>
auto std::set< _Key, _Compare, _Alloc >::lower_bound (
            const _Kt & __x) const -> decltype(const_iterator(_M_t._M_lower_bound_tr(__x)))
[inline]
```

Finds the beginning of a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

lower_bound() [3/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
iterator std::set< _Key, _Compare, _Alloc >::lower_bound (
            const key_type & __x)  [inline]
```

Finds the beginning of a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

lower_bound() [4/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
const_iterator std::set< _Key, _Compare, _Alloc >::lower_bound (
            const key_type & __x) const  [inline]
```

Finds the beginning of a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

max_size()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
size_type std::set< _Key, _Compare, _Alloc >::max_size () const  [inline], [noexcept]
```

Returns the maximum size of the set.

operator=() [1/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
set & std::set< _Key, _Compare, _Alloc >::operator= (
            const set< _Key, _Compare, _Alloc > & )  [default]
```

Set assignment operator.
Whether the allocator is copied depends on the allocator traits.

operator=() [2/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
set & std::set< _Key, _Compare, _Alloc >::operator= (
            initializer_list< value_type > __l)  [inline]
```

Set list assignment operator.

Parameters

↩ _↩ ↩ _↩ *l*	An initializer_list.

This function fills a set with copies of the elements in the initializer list *__l*.
Note that the assignment completely changes the set and that the resulting set's size is the same as the number of elements assigned.

operator=() [3/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
set & std::set< _Key, _Compare, _Alloc >::operator= (
            set< _Key, _Compare, _Alloc > && )  [default]
```

Move assignment operator.

rbegin()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
reverse_iterator std::set< _Key, _Compare, _Alloc >::rbegin () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the last element in the set. Iteration is done in descending order according to the keys.

rend()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
reverse_iterator std::set< _Key, _Compare, _Alloc >::rend () const  [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the set. Iteration is done in descending order according to the keys.

size()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
size_type std::set< _Key, _Compare, _Alloc >::size () const  [inline], [noexcept]
```

Returns the size of the set.

swap()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
void std::set< _Key, _Compare, _Alloc >::swap (
            set< _Key, _Compare, _Alloc > & __x)  [inline], [noexcept]
```

Swaps data with another set.

Parameters

_↩ _x	A set of the same element and allocator types.

This exchanges the elements between two sets in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global std::swap() function is specialized such that std::swap(s1,s2) will feed to this function.
Whether the allocators are swapped depends on the allocator traits.
Referenced by std::swap().

upper_bound() [1/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
template<typename _Kt>
auto std::set< _Key, _Compare, _Alloc >::upper_bound (
            const _Kt & __x) -> decltype(iterator(_M_t._M_upper_bound_tr(__x)))  [inline]
```

Finds the end of a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Iterator pointing to the first element greater than key, or end().

upper_bound() [2/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
template<typename _Kt>
auto std::set< _Key, _Compare, _Alloc >::upper_bound (
            const _Kt & __x) const -> decltype(const_iterator(_M_t._M_upper_bound_tr(__x)))
[inline]
```

Finds the end of a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Iterator pointing to the first element greater than key, or end().

upper_bound() [3/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
iterator std::set< _Key, _Compare, _Alloc >::upper_bound (
            const key_type & __x)  [inline]
```

Finds the end of a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Iterator pointing to the first element greater than key, or end().

upper_bound() [4/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
const_iterator std::set< _Key, _Compare, _Alloc >::upper_bound (
            const key_type & __x) const  [inline]
```

Finds the end of a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Iterator pointing to the first element greater than key, or end().

value_comp()

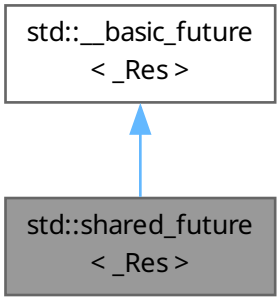
```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
value_compare std::set< _Key, _Compare, _Alloc >::value_comp () const  [inline]
```

Returns the comparison object with which the set was constructed.
The documentation for this class was generated from the following files:

- stl_multiset.h
- stl_set.h

5.886 std::shared_future< _Res > Class Template Reference

`#include <future>`
Inheritance diagram for std::shared_future< _Res >:



Public Member Functions

- shared_future (const shared_future &__sf) noexcept
- shared_future (future< _Res > &&__uf) noexcept
- shared_future (shared_future &&__sf) noexcept
- const _Res & get () const
- shared_future & **operator=** (const shared_future &__sf) noexcept
- shared_future & **operator=** (shared_future &&__sf) noexcept
- bool **valid** () const noexcept
- void **wait** () const
- template<typename _Rep, typename _Period>
  future_status **wait_for** (const chrono::duration< _Rep, _Period > &__rel) const
- template<typename _Clock, typename _Duration>
  future_status **wait_until** (const chrono::time_point< _Clock, _Duration > &__abs) const

Protected Types

- typedef __future_base::_Result< _Res > & **__result_type**
- typedef shared_ptr< _State_base > **__state_type**

Protected Member Functions

- __result_type _M_get_result () const
- void **_M_swap** (__basic_future &__that) noexcept

5.886.1 Detailed Description

template<typename _Res>
class std::shared_future< _Res >

Primary template for shared_future.

5.886.2 Constructor & Destructor Documentation

shared_future() [1/3]

```
template<typename _Res>
std::shared_future< _Res >::shared_future (
            const shared_future< _Res > & __sf)  [inline], [noexcept]
```

Copy constructor.

shared_future() [2/3]

```
template<typename _Res>
std::shared_future< _Res >::shared_future (
            future< _Res > && __uf)  [inline], [noexcept]
```

Construct from a future rvalue.

shared_future() [3/3]

```
template<typename _Res>
std::shared_future< _Res >::shared_future (
            shared_future< _Res > && __sf)  [inline], [noexcept]
```

Construct from a shared_future rvalue.

5.886.3 Member Function Documentation

_M_get_result()

```
template<typename _Res>
__result_type std::__basic_future< _Res >::_M_get_result () const  [inline], [protected], [inherited]
```

Wait for the state to be ready and rethrow any stored exception.

get()

```
template<typename _Res>
const _Res & std::shared_future< _Res >::get () const  [inline]
```

Retrieving the value.

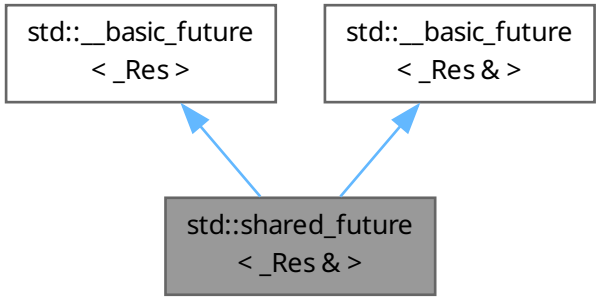
The documentation for this class was generated from the following file:

- future

5.887 std::shared_future< _Res & > Class Template Reference

`#include <future>`

Inheritance diagram for std::shared_future< _Res & >:



Public Member Functions

- shared_future (const shared_future &__sf)
- shared_future (const shared_future &__sf) noexcept
- shared_future (future< _Res & > &&__uf) noexcept
- shared_future (future< _Res > &&__uf) noexcept
- shared_future (shared_future &&__sf) noexcept
- shared_future (shared_future &&__sf) noexcept
- const _Res & get () const
- _Res & get () const
- shared_future & **operator=** (const shared_future &__sf)
- shared_future & **operator=** (const shared_future &__sf) noexcept
- shared_future & **operator=** (shared_future &&__sf) noexcept
- shared_future & **operator=** (shared_future &&__sf) noexcept
- bool **valid** () const noexcept
- void **wait** () const
- future_status **wait_for** (const chrono::duration< _Rep, _Period > &__rel) const
- future_status **wait_until** (const chrono::time_point< _Clock, _Duration > &__abs) const

Protected Types

- typedef __future_base::_Result< _Res > & **__result_type**
- typedef shared_ptr< _State_base > **__state_type**

Protected Member Functions

- __result_type _M_get_result () const
- void **_M_swap** (__basic_future &__that) noexcept

5.887.1 Detailed Description

template<typename _Res>
class std::shared_future< _Res & >

Partial specialization for shared_future<R&>

5.887.2 Constructor & Destructor Documentation

shared_future() [1/6]

```
template<typename _Res>
std::shared_future< _Res & >::shared_future (
            const shared_future< _Res & > & __sf)  [inline]
```

Copy constructor.

shared_future() [2/6]

```
template<typename _Res>
std::shared_future< _Res & >::shared_future (
            future< _Res & > && __uf)  [inline], [noexcept]
```

Construct from a future rvalue.

shared_future() [3/6]

```
template<typename _Res>
std::shared_future< _Res & >::shared_future (
            shared_future< _Res & > && __sf)  [inline], [noexcept]
```

Construct from a shared_future rvalue.

shared_future() [4/6]

```
std::shared_future< _Res >::shared_future (
            const shared_future< _Res & > & __sf)  [inline], [noexcept]
```

Copy constructor.

shared_future() [5/6]

```
std::shared_future< _Res >::shared_future (
            future< _Res > && __uf)  [inline], [noexcept]
```

Construct from a future rvalue.

shared_future() [6/6]

```
std::shared_future< _Res >::shared_future (
            shared_future< _Res & > && __sf)  [inline], [noexcept]
```

Construct from a shared_future rvalue.

5.887.3 Member Function Documentation

_M_get_result()

```
__result_type std::__basic_future< _Res >::_M_get_result () const  [inline], [protected]
```

Wait for the state to be ready and rethrow any stored exception.

get() [1/2]

```
const _Res & std::shared_future< _Res >::get () const  [inline]
```

Retrieving the value.

get() [2/2]

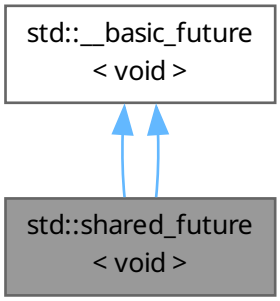
```
template<typename _Res>
_Res & std::shared_future< _Res & >::get () const  [inline]
```

Retrieving the value.
The documentation for this class was generated from the following file:

- future

5.888 std::shared_future< void > Class Reference

`#include <future>`
Inheritance diagram for std::shared_future< void >:



Public Member Functions

- shared_future (const shared_future &__sf)
- shared_future (const shared_future &__sf) noexcept
- shared_future (future< void > &&__uf) noexcept
- shared_future (future< void > &&__uf) noexcept
- shared_future (shared_future &&__sf) noexcept
- shared_future (shared_future &&__sf) noexcept
- const void & get () const
- void **get** () const
- shared_future & **operator=** (const shared_future &__sf)
- shared_future & **operator=** (const shared_future &__sf) noexcept
- shared_future & **operator=** (shared_future &&__sf) noexcept
- shared_future & **operator=** (shared_future &&__sf) noexcept
- bool **valid** () const noexcept
- void **wait** () const
- future_status **wait_for** (const chrono::duration< _Rep, _Period > &__rel) const
- future_status **wait_until** (const chrono::time_point< _Clock, _Duration > &__abs) const

Protected Types

- typedef __future_base::_Result< void > & **__result_type**
- typedef shared_ptr< _State_base > **__state_type**

Protected Member Functions

- __result_type _M_get_result () const
- void **_M_swap** (__basic_future &__that) noexcept

5.888.1 Detailed Description

Explicit specialization for shared_future<void>

5.888.2 Constructor & Destructor Documentation

shared_future() [1/6]

```
std::shared_future< void >::shared_future (
            const shared_future< void > & __sf)  [inline]
```

Copy constructor.

shared_future() [2/6]

```
std::shared_future< void >::shared_future (
            future< void > && __uf)  [inline], [noexcept]
```

Construct from a future rvalue.

shared_future() [3/6]

```
std::shared_future< void >::shared_future (
            shared_future< void > && __sf)  [inline], [noexcept]
```

Construct from a shared_future rvalue.

shared_future() [4/6]

```
std::shared_future< void >::shared_future (
            const shared_future< void > & __sf)  [inline], [noexcept]
```

Copy constructor.

shared_future() [5/6]

```
std::shared_future< void >::shared_future (
            future< void > && __uf)  [inline], [noexcept]
```

Construct from a future rvalue.

shared_future() [6/6]

```
std::shared_future< void >::shared_future (
            shared_future< void > && __sf)  [inline], [noexcept]
```

Construct from a shared_future rvalue.

5.888.3 Member Function Documentation

_M_get_result()

```
__result_type std::__basic_future< void >::_M_get_result () const  [inline], [protected]
```

Wait for the state to be ready and rethrow any stored exception.

get()

```
const void & std::shared_future< void >::get () const  [inline]
```

Retrieving the value.

The documentation for this class was generated from the following file:

- future

5.889 std::shared_lock< _Mutex > Class Template Reference

```
#include <shared_mutex>
```

Public Types

- typedef _Mutex **mutex_type**

Public Member Functions

- **shared_lock** (mutex_type &__m)
- **shared_lock** (mutex_type &__m, adopt_lock_t)
- template<typename _Rep, typename _Period>
  shared_lock (mutex_type &__m, const chrono::duration< _Rep, _Period > &__rel_time)
- template<typename _Clock, typename _Duration>
  shared_lock (mutex_type &__m, const chrono::time_point< _Clock, _Duration > &__abs_time)
- **shared_lock** (mutex_type &__m, defer_lock_t) noexcept
- **shared_lock** (mutex_type &__m, try_to_lock_t)
- **shared_lock** (shared_lock &&__sl) noexcept
- **shared_lock** (shared_lock const &)=delete
- void **lock** ()
- mutex_type * **mutex** () const noexcept
- **operator bool** () const noexcept
- shared_lock & **operator=** (shared_lock &&__sl) noexcept
- shared_lock & **operator=** (shared_lock const &)=delete
- bool **owns_lock** () const noexcept
- mutex_type * **release** () noexcept
- void **swap** (shared_lock &__u) noexcept
- bool **try_lock** ()
- template<typename _Rep, typename _Period>
  bool **try_lock_for** (const chrono::duration< _Rep, _Period > &__rel_time)
- template<typename _Clock, typename _Duration>
  bool **try_lock_until** (const chrono::time_point< _Clock, _Duration > &__abs_time)
- void **unlock** ()

5.889.1 Detailed Description

template<typename _Mutex>
class std::shared_lock< _Mutex >

shared_lock
The documentation for this class was generated from the following file:

- shared_mutex

5.890 std::shared_ptr< _Tp > Class Template Reference

`#include <memory>`

Public Types

- using element_type
- using **weak_type**

Public Member Functions

- constexpr shared_ptr () noexcept
- template<typename _Yp, typename = _Constructible<_Yp*>>
  shared_ptr (_Yp *__p)
- template<typename _Yp, typename _Deleter, typename = _Constructible<_Yp*, _Deleter>>
  shared_ptr (_Yp *__p, _Deleter __d)
- template<typename _Yp, typename _Deleter, typename _Alloc, typename = _Constructible<_Yp*, _Deleter, _Alloc>>
  shared_ptr (_Yp *__p, _Deleter __d, _Alloc __a)
- template<typename _Yp, typename = _Constructible<auto_ptr<_Yp>>>
  shared_ptr (auto_ptr< _Yp > &&__r)
- shared_ptr (const shared_ptr &) noexcept=default
- template<typename _Yp, typename = _Constructible<const shared_ptr<_Yp>&>>
  shared_ptr (const shared_ptr< _Yp > &__r) noexcept
- template<typename _Yp>
  shared_ptr (const shared_ptr< _Yp > &__r, element_type *__p) noexcept
- template<typename _Yp, typename = _Constructible<const weak_ptr<_Yp>&>>
  shared_ptr (const weak_ptr< _Yp > &__r)
- template<typename _Deleter>
  shared_ptr (nullptr_t __p, _Deleter __d)
- template<typename _Deleter, typename _Alloc>
  shared_ptr (nullptr_t __p, _Deleter __d, _Alloc __a)
- constexpr shared_ptr (nullptr_t) noexcept
- shared_ptr (shared_ptr &&__r) noexcept
- template<typename _Yp, typename = _Constructible<shared_ptr<_Yp>>>
  shared_ptr (shared_ptr< _Yp > &&__r) noexcept
- template<typename _Yp>
  shared_ptr (shared_ptr< _Yp > &&__r, element_type *__p) noexcept
- template<typename _Tp1, typename>
  shared_ptr (std::auto_ptr< _Tp1 > &&__r)
- template<typename _Yp, typename _Del, typename = _Constructible<unique_ptr<_Yp, _Del>>>
  shared_ptr (unique_ptr< _Yp, _Del > &&__r)
- element_type * get () const noexcept
- operator bool () const noexcept
- element_type & **operator*** () const noexcept

- element_type * **operator->** () const noexcept
- template<typename _Yp>
  _Assignable< auto_ptr< _Yp > > **operator=** (auto_ptr< _Yp > &&__r)
- shared_ptr & **operator=** (const shared_ptr &) noexcept=default
- template<typename _Yp>
  _Assignable< const shared_ptr< _Yp > & > **operator=** (const shared_ptr< _Yp > &__r) noexcept
- shared_ptr & **operator=** (shared_ptr &&__r) noexcept
- template<class _Yp>
  _Assignable< shared_ptr< _Yp > > **operator=** (shared_ptr< _Yp > &&__r) noexcept
- template<typename _Yp, typename _Del>
  _Assignable< unique_ptr< _Yp, _Del > > **operator=** (unique_ptr< _Yp, _Del > &&__r)
- void **reset** () noexcept
- template<typename _Yp>
  _SafeConv< _Yp > **reset** (_Yp *__p)
- template<typename _Yp, typename _Deleter>
  _SafeConv< _Yp > **reset** (_Yp *__p, _Deleter __d)
- template<typename _Yp, typename _Deleter, typename _Alloc>
  _SafeConv< _Yp > **reset** (_Yp *__p, _Deleter __d, _Alloc __a)
- void swap (__shared_ptr< _Tp, _Lp > &__other) noexcept
- bool unique () const noexcept
- long use_count () const noexcept

- template<typename _Tp1>
  bool owner_before (__shared_ptr< _Tp1, _Lp > const &__rhs) const noexcept
- template<typename _Tp1>
  bool owner_before (__weak_ptr< _Tp1, _Lp > const &__rhs) const noexcept

Friends

- template<typename _Yp, typename _Alloc, typename... _Args>
  shared_ptr< _NonArray< _Yp > > **allocate_shared** (const _Alloc &, _Args &&...)
- template<typename _Yp, typename... _Args>
  shared_ptr< _NonArray< _Yp > > **make_shared** (_Args &&...)
- class **weak_ptr< _Tp >**

Related Symbols

(Note that these are not member symbols.)

- template<typename _Del, typename _Tp>
  _Del * get_deleter (const shared_ptr< _Tp > &__p) noexcept
- template<typename _Ch, typename _Tr, typename _Tp, _Lock_policy _Lp>
  std::basic_ostream< _Ch, _Tr > & operator<< (std::basic_ostream< _Ch, _Tr > &__os, const __shared_ptr< _Tp, _Lp > &__p)

- template<typename _Tp, typename _Up>
  bool operator== (const shared_ptr< _Tp > &__a, const shared_ptr< _Up > &__b) noexcept
- template<typename _Tp>
  bool operator== (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp>
  void swap (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b) noexcept
- template<typename _Tp, typename _Up>
  shared_ptr< _Tp > static_pointer_cast (const shared_ptr< _Up > &__r) noexcept

- template<typename _Tp, typename _Up>
  shared_ptr< _Tp > const_pointer_cast (const shared_ptr< _Up > &__r) noexcept
- template<typename _Tp, typename _Up>
  shared_ptr< _Tp > dynamic_pointer_cast (const shared_ptr< _Up > &__r) noexcept
- template<typename _Tp, typename _Up>
  shared_ptr< _Tp > reinterpret_pointer_cast (const shared_ptr< _Up > &__r) noexcept
- template<typename _Tp, typename _Up>
  shared_ptr< _Tp > static_pointer_cast (shared_ptr< _Up > &&__r) noexcept
- template<typename _Tp, typename _Up>
  shared_ptr< _Tp > const_pointer_cast (shared_ptr< _Up > &&__r) noexcept
- template<typename _Tp, typename _Up>
  shared_ptr< _Tp > dynamic_pointer_cast (shared_ptr< _Up > &&__r) noexcept
- template<typename _Tp, typename _Up>
  shared_ptr< _Tp > reinterpret_pointer_cast (shared_ptr< _Up > &&__r) noexcept

- template<typename _Tp, typename _Alloc, typename... _Args>
  shared_ptr< _NonArray< _Tp > > allocate_shared (const _Alloc &__a, _Args &&... __args)
- template<typename _Tp, typename... _Args>
  shared_ptr< _NonArray< _Tp > > make_shared (_Args &&... __args)

5.890.1 Detailed Description

template<typename _Tp>
class std::shared_ptr< _Tp >

A smart pointer with reference-counted copy semantics.

Since

C++11

A `shared_ptr` object is either empty or *owns* a pointer passed to the constructor. Copies of a `shared_ptr` share ownership of the same pointer. When the last `shared_ptr` that owns the pointer is destroyed or reset, the owned pointer is freed (either by `delete` or by invoking a custom deleter that was passed to the constructor).
A `shared_ptr` also stores another pointer, which is usually (but not always) the same pointer as it owns. The stored pointer can be retrieved by calling the `get()` member function.
The equality and relational operators for `shared_ptr` only compare the stored pointer returned by `get()`, not the owned pointer. To test whether two `shared_ptr` objects share ownership of the same pointer see `std::shared_ptr::owner_before` and `std::owner_less`.

5.890.2 Member Typedef Documentation

element_type

```
template<typename _Tp>
using std::shared_ptr< _Tp >::element_type
```

The type pointed to by the stored pointer, remove_extent_t<_Tp>

5.890.3 Constructor & Destructor Documentation

shared_ptr() [1/14]

```
template<typename _Tp>
std::shared_ptr< _Tp >::shared_ptr ()  [inline], [constexpr], [noexcept]
```

Construct an empty shared_ptr.

Postcondition

use_count()==0 && get()==0

shared_ptr() [2/14]

```
template<typename _Tp>
std::shared_ptr< _Tp >::shared_ptr (
            const shared_ptr< _Tp > & )  [default], [noexcept]
```

Copy constructor.

shared_ptr() [3/14]

```
template<typename _Tp>
template<typename _Yp, typename = _Constructible<_Yp*>>
std::shared_ptr< _Tp >::shared_ptr (
            _Yp * __p)  [inline], [explicit]
```

Construct a shared_ptr that owns the pointer *__p*.

Parameters

_↩ _p	A pointer that is convertible to element_type*.

Postcondition

use_count() == 1 && get() == __p

Exceptions

std::bad_alloc,in	which case `delete` *__p* is called.

shared_ptr() [4/14]

```
template<typename _Tp>
template<typename _Yp, typename _Deleter, typename = _Constructible<_Yp*, _Deleter>>
std::shared_ptr< _Tp >::shared_ptr (
            _Yp * __p,
            _Deleter __d)  [inline]
```

Construct a shared_ptr that owns the pointer *__p* and the deleter *__d*.

Parameters

_↩ _p	A pointer.
_↩ _d	A deleter.

Postcondition

use_count() == 1 && get() == __p

Exceptions

std::bad_alloc,in	which case *__d(__p)* is called.

Requirements: _Deleter's copy constructor and destructor must not throw
__shared_ptr will release __p by calling __d(__p)

shared_ptr() [5/14]

```
template<typename _Tp>
template<typename _Deleter>
std::shared_ptr< _Tp >::shared_ptr (
            nullptr_t __p,
            _Deleter __d)  [inline]
```

Construct a shared_ptr that owns a null pointer and the deleter *__d*.

Parameters

_↩ _p	A null pointer constant.
_↩ _d	A deleter.

Postcondition

use_count() == 1 && get() == __p

Exceptions

std::bad_alloc,in	which case *__d(__p)* is called.

Requirements: _Deleter's copy constructor and destructor must not throw
The last owner will call __d(__p)

shared_ptr() [6/14]

```
template<typename _Tp>
template<typename _Yp, typename _Deleter, typename _Alloc, typename = _Constructible<_Yp*, _↩
Deleter, _Alloc>>
std::shared_ptr< _Tp >::shared_ptr (
            _Yp * __p,
            _Deleter __d,
            _Alloc __a)  [inline]
```

Construct a shared_ptr that owns the pointer *__p* and the deleter *__d*.

Parameters

_↩ _p	A pointer.
_↩ _d	A deleter.
_↩ _a	An allocator.

Postcondition

use_count() == 1 && get() == __p

Exceptions

std::bad_alloc,in	which case *__d(__p)* is called.

Requirements: _Deleter's copy constructor and destructor must not throw _Alloc's copy constructor and destructor must not throw.
__shared_ptr will release __p by calling __d(__p)

shared_ptr() [7/14]

```
template<typename _Tp>
template<typename _Deleter, typename _Alloc>
std::shared_ptr< _Tp >::shared_ptr (
            nullptr_t __p,
            _Deleter __d,
            _Alloc __a)  [inline]
```

Construct a shared_ptr that owns a null pointer and the deleter *__d*.

Parameters

_↩ _p	A null pointer constant.
_↩ _d	A deleter.
_↩ _a	An allocator.

Postcondition

use_count() == 1 && get() == __p

Exceptions

std::bad_alloc,in	which case *__d(__p)* is called.

Requirements: _Deleter's copy constructor and destructor must not throw _Alloc's copy constructor and destructor must not throw.
The last owner will call __d(__p)

shared_ptr() [8/14]

```
template<typename _Tp>
template<typename _Yp>
std::shared_ptr< _Tp >::shared_ptr (
            const shared_ptr< _Yp > & __r,
            element_type * __p)  [inline], [noexcept]
```

Constructs a `shared_ptr` instance that stores `__p` and shares ownership with `__r`.

Parameters

_↩ _r	A shared_ptr.
_↩ _p	A pointer that will remain valid while *__r is valid.

Postcondition

```
get() == __p && use_count() == __r.use_count()
```

This can be used to construct a shared_ptr to a sub-object of an object managed by an existing shared_ptr. The complete object will remain valid while any shared_ptr owns it, even if they don't store a pointer to the complete object.

```
shared_ptr<pair<int,int>> pii(new pair<int,int>());
shared_ptr<int> pi(pii, &pii->first);
assert(pii.use_count() == 2);
```

shared_ptr() [9/14]

```
template<typename _Tp>
template<typename _Yp>
std::shared_ptr< _Tp >::shared_ptr (
            shared_ptr< _Yp > && __r,
            element_type * __p)  [inline], [noexcept]
```

Constructs a shared_ptr instance that stores __p and shares ownership with __r.

Parameters

_↩ _r	A shared_ptr.
_↩ _p	A pointer that will remain valid while *__r is valid.

Postcondition

```
get() == __p && !__r.use_count() && !__r.get()
```

Since

C++17

This can be used to construct a shared_ptr to a sub-object of an object managed by an existing shared_ptr. The complete object will remain valid while any shared_ptr owns it, even if they don't store a pointer to the complete object.

```
shared_ptr<pair<int,int>> pii(new pair<int,int>());
shared_ptr<int> pi1(pii, &pii->first);
assert(pii.use_count() == 2);
shared_ptr<int> pi2(std::move(pii), &pii->second);
assert(pii.use_count() == 0);
```

shared_ptr() [10/14]

```
template<typename _Tp>
template<typename _Yp, typename = _Constructible<const shared_ptr<_Yp>&>>
std::shared_ptr< _Tp >::shared_ptr (
            const shared_ptr< _Yp > & __r)  [inline], [noexcept]
```

If __r is empty, constructs an empty shared_ptr; otherwise construct a shared_ptr that shares ownership with __r.

Parameters

↩ _↩ ↩ _↩ *r*	A shared_ptr.

Postcondition

get() == __r.get() && use_count() == __r.use_count()

shared_ptr() [11/14]

```
template<typename _Tp>
std::shared_ptr< _Tp >::shared_ptr (
            shared_ptr< _Tp > && __r)  [inline], [noexcept]
```

Move-constructs a shared_ptr instance from *__r*.

Parameters

↩ _↩ ↩ _↩ *r*	A shared_ptr rvalue.

Postcondition

∗this contains the old value of *__r*, *__r* is empty.

shared_ptr() [12/14]

```
template<typename _Tp>
template<typename _Yp, typename = _Constructible<shared_ptr<_Yp>>>
std::shared_ptr< _Tp >::shared_ptr (
            shared_ptr< _Yp > && __r)  [inline], [noexcept]
```

Move-constructs a shared_ptr instance from *__r*.

Parameters

↩ _↩ ↩ _↩ *r*	A shared_ptr rvalue.

Postcondition

∗this contains the old value of *__r*, *__r* is empty.

shared_ptr() [13/14]

```
template<typename _Tp>
template<typename _Yp, typename = _Constructible<const weak_ptr<_Yp>&>>
std::shared_ptr< _Tp >::shared_ptr (
            const weak_ptr< _Yp > & __r)  [inline], [explicit]
```

Constructs a shared_ptr that shares ownership with *__r* and stores a copy of the pointer stored in *__r*.

Parameters

↩ _↩ ↩ _↩ *r*	A weak_ptr.

Postcondition

use_count() == __r.use_count()

Exceptions

bad_weak_ptr	when __r.expired(), in which case the constructor has no effect.

shared_ptr() [14/14]

```
template<typename _Tp>
std::shared_ptr< _Tp >::shared_ptr (
            nullptr_t )  [inline], [constexpr], [noexcept]
```

Construct an empty shared_ptr.

Postcondition

use_count() == 0 && get() == nullptr

5.890.4 Member Function Documentation

get()

```
template<typename _Tp, _Lock_policy _Lp>
element_type * std::__shared_ptr< _Tp, _Lp >::get () const  [inline], [noexcept], [inherited]
```

Return the stored pointer.

operator bool()

```
template<typename _Tp, _Lock_policy _Lp>
std::__shared_ptr< _Tp, _Lp >::operator bool () const  [inline], [explicit], [noexcept], [inherited]
```

Return true if the stored pointer is not null.

owner_before() [1/2]

```
template<typename _Tp, _Lock_policy _Lp>
template<typename _Tp1>
bool std::__shared_ptr< _Tp, _Lp >::owner_before (
            __shared_ptr< _Tp1, _Lp > const & __rhs) const  [inline], [noexcept], [inherited]
```

Define an ordering based on ownership.
This function defines a strict weak ordering between two shared_ptr or weak_ptr objects, such that one object is less than the other unless they share ownership of the same pointer, or are both empty.

owner_before() [2/2]

```
template<typename _Tp, _Lock_policy _Lp>
template<typename _Tp1>
bool std::__shared_ptr< _Tp, _Lp >::owner_before (
            __weak_ptr< _Tp1, _Lp > const & __rhs) const  [inline], [noexcept], [inherited]
```

Define an ordering based on ownership.
This function defines a strict weak ordering between two shared_ptr or weak_ptr objects, such that one object is less than the other unless they share ownership of the same pointer, or are both empty.

swap()

```
template<typename _Tp, _Lock_policy _Lp>
void std::__shared_ptr< _Tp, _Lp >::swap (
            __shared_ptr< _Tp, _Lp > & __other)  [inline], [noexcept], [inherited]
```

Exchange both the owned pointer and the stored pointer.

unique()

```
template<typename _Tp, _Lock_policy _Lp>
bool std::__shared_ptr< _Tp, _Lp >::unique () const  [inline], [noexcept], [inherited]
```

Return true if use_count() == 1.

use_count()

```
template<typename _Tp, _Lock_policy _Lp>
long std::__shared_ptr< _Tp, _Lp >::use_count () const  [inline], [noexcept], [inherited]
```

If ∗this owns a pointer, return the number of owners, otherwise zero.
The documentation for this class was generated from the following files:

- bits/shared_ptr.h
- auto_ptr.h

5.891 std::shared_timed_mutex Class Reference

```
#include <shared_mutex>
```

Public Member Functions

- **shared_timed_mutex** (const shared_timed_mutex &)=delete
- void **lock** ()
- void **lock_shared** ()
- shared_timed_mutex & **operator=** (const shared_timed_mutex &)=delete
- bool **try_lock** ()
- template<typename _Rep, typename _Period>
  bool **try_lock_for** (const chrono::duration< _Rep, _Period > &__rtime)
- bool **try_lock_shared** ()
- template<typename _Rep, typename _Period>
  bool **try_lock_shared_for** (const chrono::duration< _Rep, _Period > &__rtime)
- template<typename _Clock, typename _Duration>
  bool **try_lock_shared_until** (const chrono::time_point< _Clock, _Duration > &__abs_time)

- template<typename _Clock, typename _Duration>
  bool **try_lock_until** (const chrono::time_point< _Clock, _Duration > &__abs_time)
- void **unlock** ()
- void **unlock_shared** ()

5.891.1 Detailed Description

The standard shared timed mutex type.
The documentation for this class was generated from the following file:

- shared_mutex

5.892 std::shuffle_order_engine< _RandomNumberEngine, __k > Class Template Reference

`#include <random>`

Public Types

- template<typename _Sseq>
  using **_If_seed_seq**
- typedef _RandomNumberEngine::result_type result_type

Public Member Functions

- shuffle_order_engine ()
- shuffle_order_engine (_RandomNumberEngine &&__rng)
- template<typename _Sseq, typename = _If_seed_seq<_Sseq>>
  shuffle_order_engine (_Sseq &__q)
- shuffle_order_engine (const _RandomNumberEngine &__rng)
- shuffle_order_engine (result_type __s)
- const _RandomNumberEngine & base () const noexcept
- void discard (unsigned long long __z)
- result_type operator() ()
- void seed ()
- template<typename _Sseq>
  _If_seed_seq< _Sseq > seed (_Sseq &__q)
- void seed (result_type __s)

Static Public Member Functions

- static constexpr result_type max ()
- static constexpr result_type min ()

Static Public Attributes

- static constexpr size_t **table_size**

Friends

- template<typename _RandomNumberEngine1, size_t __k1, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::shuffle_order_engine< _RandomNumberEngine1, __k1 > &__x)
- bool operator== (const shuffle_order_engine &__lhs, const shuffle_order_engine &__rhs)
- template<typename _RandomNumberEngine1, size_t __k1, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::shuffle_order_engine< _RandomNumberEngine1, __k1 > &__x)

5.892.1 Detailed Description

template<typename _RandomNumberEngine, size_t __k>
class std::shuffle_order_engine< _RandomNumberEngine, __k >

Produces random numbers by reordering random numbers from some base engine.
The values from the base engine are stored in a sequence of size `__k` and shuffled by an algorithm that depends on those values.

Since

C++11

5.892.2 Member Typedef Documentation

result_type

```
template<typename _RandomNumberEngine, size_t __k>
typedef _RandomNumberEngine::result_type std::shuffle_order_engine< _RandomNumberEngine, __k >↩
::result_type
```

The type of the generated random value.

5.892.3 Constructor & Destructor Documentation

shuffle_order_engine() [1/5]

```
template<typename _RandomNumberEngine, size_t __k>
std::shuffle_order_engine< _RandomNumberEngine, __k >::shuffle_order_engine ()  [inline]
```

Constructs a default shuffle_order_engine engine.
The underlying engine is default constructed as well.

shuffle_order_engine() [2/5]

```
template<typename _RandomNumberEngine, size_t __k>
std::shuffle_order_engine< _RandomNumberEngine, __k >::shuffle_order_engine (
            const _RandomNumberEngine & __rng)  [inline], [explicit]
```

Copy constructs a shuffle_order_engine engine.
Copies an existing base class random number generator.

Parameters

__rng	An existing (base class) engine object.

shuffle_order_engine() [3/5]

```
template<typename _RandomNumberEngine, size_t __k>
std::shuffle_order_engine< _RandomNumberEngine, __k >::shuffle_order_engine (
            _RandomNumberEngine && __rng)  [inline], [explicit]
```

Move constructs a shuffle_order_engine engine.
Copies an existing base class random number generator.

Parameters

__rng	An existing (base class) engine object.

shuffle_order_engine() [4/5]

```
template<typename _RandomNumberEngine, size_t __k>
std::shuffle_order_engine< _RandomNumberEngine, __k >::shuffle_order_engine (
            result_type __s)  [inline], [explicit]
```

Seed constructs a shuffle_order_engine engine.
Constructs the underlying generator engine seeded with __s.

Parameters

_↩ _s	A seed value for the base class engine.

shuffle_order_engine() [5/5]

```
template<typename _RandomNumberEngine, size_t __k>
template<typename _Sseq, typename = _If_seed_seq<_Sseq>>
std::shuffle_order_engine< _RandomNumberEngine, __k >::shuffle_order_engine (
            _Sseq & __q)  [inline], [explicit]
```

Generator construct a shuffle_order_engine engine.

Parameters

_↩ _q	A seed sequence.

5.892.4 Member Function Documentation

base()

```
template<typename _RandomNumberEngine, size_t __k>
const _RandomNumberEngine & std::shuffle_order_engine< _RandomNumberEngine, __k >::base () const
[inline], [noexcept]
```

Gets a const reference to the underlying generator engine object.

discard()

```
template<typename _RandomNumberEngine, size_t __k>
void std::shuffle_order_engine< _RandomNumberEngine, __k >::discard (
            unsigned long long __z)  [inline]
```

Discard a sequence of random numbers.

max()

```
template<typename _RandomNumberEngine, size_t __k>
static constexpr result_type std::shuffle_order_engine< _RandomNumberEngine, __k >::max ()  [inline],
[static], [constexpr]
```

Gets the maximum value in the generated random number range.

min()

```
template<typename _RandomNumberEngine, size_t __k>
static constexpr result_type std::shuffle_order_engine< _RandomNumberEngine, __k >::min ()  [inline],
[static], [constexpr]
```

Gets the minimum value in the generated random number range.

operator()()

```
template<typename _RandomNumberEngine, size_t __k>
shuffle_order_engine< _RandomNumberEngine, __k >::result_type std::shuffle_order_engine< _↩
RandomNumberEngine, __k >::operator() ()
```

Gets the next value in the generated random number sequence.
References std::max(), and std::min().

seed() [1/3]

```
template<typename _RandomNumberEngine, size_t __k>
void std::shuffle_order_engine< _RandomNumberEngine, __k >::seed ()  [inline]
```

Reseeds the shuffle_order_engine object with the default seed for the underlying base class generator engine.

seed() [2/3]

```
template<typename _RandomNumberEngine, size_t __k>
template<typename _Sseq>
_If_seed_seq< _Sseq > std::shuffle_order_engine< _RandomNumberEngine, __k >::seed (
            _Sseq & __q)  [inline]
```

Reseeds the shuffle_order_engine object with the given seed sequence.

Parameters

_↩ _q	A seed generator function.

seed() [3/3]

```
template<typename _RandomNumberEngine, size_t __k>
void std::shuffle_order_engine< _RandomNumberEngine, __k >::seed (
            result_type __s)  [inline]
```

Reseeds the shuffle_order_engine object with the default seed for the underlying base class generator engine.

5.892.5 Friends And Related Symbol Documentation

operator<<

```
template<typename _RandomNumberEngine, size_t __k>
template<typename _RandomNumberEngine1, size_t __k1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
```

```
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::shuffle_order_engine< _RandomNumberEngine1, __k1 > & __x)  [friend]
```

Inserts the current state of a shuffle_order_engine random number generator engine __x into the output stream __os.

Parameters

__os	An output stream.
__x	A shuffle_order_engine random number generator engine.

Returns

The output stream with the state of __x inserted or in an error state.

operator==

```
template<typename _RandomNumberEngine, size_t __k>
bool operator== (
            const shuffle_order_engine< _RandomNumberEngine, __k > & __lhs,
            const shuffle_order_engine< _RandomNumberEngine, __k > & __rhs)  [friend]
```

Compares two shuffle_order_engine random number generator objects of the same type for equality.

Parameters

__lhs	A shuffle_order_engine random number generator object.
__rhs	Another shuffle_order_engine random number generator object.

Returns

true if the infinite sequences of generated values would be equal, false otherwise.

operator>>

```
template<typename _RandomNumberEngine, size_t __k>
template<typename _RandomNumberEngine1, size_t __k1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            std::shuffle_order_engine< _RandomNumberEngine1, __k1 > & __x)  [friend]
```

Extracts the current state of a % subtract_with_carry_engine random number generator engine __x from the input stream __is.

Parameters

_↩ _is	An input stream.
_↩ _x	A shuffle_order_engine random number generator engine.

Returns

The input stream with the state of __x extracted or in an error state.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

5.893 std::slice Class Reference

`#include <slice_array.h>`

Public Member Functions

- slice ()
- slice (size_t __o, size_t __d, size_t __s)
- size_t size () const
- size_t start () const
- size_t stride () const

Friends

- bool operator== (const slice &, const slice &)=default

5.893.1 Detailed Description

Class defining one-dimensional subset of an array.
The slice class represents a one-dimensional subset of an array, specified by three parameters: start offset, size, and stride. The start offset is the index of the first element of the array that is part of the subset. The size is the total number of elements in the subset. Stride is the distance between each successive array element to include in the subset.
For example, with an array of size 10, and a slice with offset 1, size 3 and stride 2, the subset consists of array elements 1, 3, and 5.

5.893.2 Friends And Related Symbol Documentation

operator==

```
bool operator== (
            const slice & ,
            const slice & )  [friend]
```

Equality comparison.
References slice().
The documentation for this class was generated from the following file:

- slice_array.h

5.894 std::slice_array< _Tp > Class Template Reference

`#include <slice_array.h>`

Public Types

- typedef _Tp **value_type**

Public Member Functions

- slice_array (const slice_array &)
- template<class _Dom>
  void **operator%=** (const _Expr< _Dom, _Tp > &) const
- void operator%= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator&=** (const _Expr< _Dom, _Tp > &) const
- void operator&= (const valarray< _Tp > &) const

- template<class _Dom>
  void **operator*=** (const _Expr< _Dom, _Tp > &) const
- void operator*= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator+=** (const _Expr< _Dom, _Tp > &) const
- void operator+= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator-=** (const _Expr< _Dom, _Tp > &) const
- void operator-= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator/=** (const _Expr< _Dom, _Tp > &) const
- void operator/= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator<<=** (const _Expr< _Dom, _Tp > &) const
- void operator<<= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator=** (const _Expr< _Dom, _Tp > &) const
- void operator= (const _Tp &) const
- slice_array & operator= (const slice_array &)
- void operator= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator>>=** (const _Expr< _Dom, _Tp > &) const
- void operator>>= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator^=** (const _Expr< _Dom, _Tp > &) const
- void operator^= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator|=** (const _Expr< _Dom, _Tp > &) const
- void operator|= (const valarray< _Tp > &) const

Friends

- class **valarray< _Tp >**

5.894.1 Detailed Description

template<typename _Tp>
class std::slice_array< _Tp >

Reference to one-dimensional subset of an array.
A slice_array is a reference to the actual elements of an array specified by a slice. The way to get a slice_array is to call operator[ ](slice) on a valarray. The returned slice_array then permits carrying operations out on the referenced subset of elements in the original valarray. For example, operator+=(valarray) will add values to the subset of elements in the underlying valarray this slice_array refers to.

Parameters

Tp	Element type.

5.894.2 Member Function Documentation

operator%=()

```
template<typename _Tp>
void std::slice_array< _Tp >::operator%= (
            const valarray< _Tp > & ) const
```

Modulo slice elements by corresponding elements of *v*.

operator&=()

```
template<typename _Tp>
void std::slice_array< _Tp >::operator&= (
            const valarray< _Tp > & ) const
```

Logical and slice elements with corresponding elements of *v*.

operator*=()

```
template<typename _Tp>
void std::slice_array< _Tp >::operator*= (
            const valarray< _Tp > & ) const
```

Multiply slice elements by corresponding elements of *v*.

operator+=()

```
template<typename _Tp>
void std::slice_array< _Tp >::operator+= (
            const valarray< _Tp > & ) const
```

Add corresponding elements of *v* to slice elements.

operator-=()

```
template<typename _Tp>
void std::slice_array< _Tp >::operator-= (
            const valarray< _Tp > & ) const
```

Subtract corresponding elements of *v* from slice elements.

operator/=()

```
template<typename _Tp>
void std::slice_array< _Tp >::operator/= (
            const valarray< _Tp > & ) const
```

Divide slice elements by corresponding elements of *v*.

operator<<=()

```
template<typename _Tp>
void std::slice_array< _Tp >::operator<<= (
            const valarray< _Tp > & ) const
```

Left shift slice elements by corresponding elements of *v*.

operator>>=()

```
template<typename _Tp>
void std::slice_array< _Tp >::operator>>= (
            const valarray< _Tp > & ) const
```

Right shift slice elements by corresponding elements of *v*.
References slice_array().

operator^=()

```
template<typename _Tp>
void std::slice_array< _Tp >::operator^= (
            const valarray< _Tp > & ) const
```

Logical xor slice elements with corresponding elements of *v*.

operator" | =()

```
template<typename _Tp>
void std::slice_array< _Tp >::operator|= (
            const valarray< _Tp > & ) const
```

Logical or slice elements with corresponding elements of *v*.
The documentation for this class was generated from the following files:

- valarray
- slice_array.h

5.895 __gnu_cxx::slist< _Tp, _Alloc > Class Template Reference

```
#include <slist>
```

Public Types

- typedef _Base::allocator_type **allocator_type**
- typedef _Slist_iterator< _Tp, const _Tp &, const _Tp * > **const_iterator**
- typedef const value_type * **const_pointer**
- typedef const value_type & **const_reference**
- typedef std::ptrdiff_t **difference_type**
- typedef _Slist_iterator< _Tp, _Tp &, _Tp * > **iterator**
- typedef value_type * **pointer**
- typedef value_type & **reference**
- typedef std::size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- template<class _InputIterator>
  slist (_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type())
- **slist** (const allocator_type &__a=allocator_type())
- **slist** (const slist &__x)
- **slist** (size_type __n)
- **slist** (size_type __n, const value_type &__x, const allocator_type &__a=allocator_type())
- template<class _InputIterator>
  void **_M_assign_dispatch** (_InputIterator __first, _InputIterator __last, std::__false_type)
- template<class _Integer>
  void **_M_assign_dispatch** (_Integer __n, _Integer __val, std::__true_type)
- void **_M_fill_assign** (size_type __n, const _Tp &__val)
- template<class _InputIterator>
  void **assign** (_InputIterator __first, _InputIterator __last)
- void **assign** (size_type __n, const _Tp &__val)

- iterator **before_begin** ()
- const_iterator **before_begin** () const
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- iterator **erase** (iterator __first, iterator __last)
- iterator **erase** (iterator __pos)
- iterator **erase_after** (iterator __before_first, iterator __last)
- iterator **erase_after** (iterator __pos)
- reference **front** ()
- const_reference **front** () const
- allocator_type **get_allocator** () const
- iterator **insert** (iterator __pos)
- template<class _InIterator>
  void **insert** (iterator __pos, _InIterator __first, _InIterator __last)
- iterator **insert** (iterator __pos, const value_type &__x)
- void **insert** (iterator __pos, size_type __n, const value_type &__x)
- iterator **insert_after** (iterator __pos)
- template<class _InIterator>
  void **insert_after** (iterator __pos, _InIterator __first, _InIterator __last)
- iterator **insert_after** (iterator __pos, const value_type &__x)
- void **insert_after** (iterator __pos, size_type __n, const value_type &__x)
- size_type **max_size** () const
- template<class _StrictWeakOrdering>
  void **merge** (slist &, _StrictWeakOrdering)
- void **merge** (slist &__x)
- slist & **operator=** (const slist &__x)
- void **pop_front** ()
- iterator **previous** (const_iterator __pos)
- const_iterator **previous** (const_iterator __pos) const
- void **push_front** ()
- void **push_front** (const value_type &__x)
- void **remove** (const _Tp &__val)
- template<class _Predicate>
  void **remove_if** (_Predicate __pred)
- void **resize** (size_type new_size)
- void **resize** (size_type new_size, const _Tp &__x)
- void **reverse** ()
- size_type **size** () const
- void **sort** ()
- template<class _StrictWeakOrdering>
  void **sort** (_StrictWeakOrdering __comp)
- void **splice** (iterator __pos, slist &__x)
- void **splice** (iterator __pos, slist &__x, iterator __first, iterator __last)
- void **splice** (iterator __pos, slist &__x, iterator __i)
- void **splice_after** (iterator __pos, iterator __before_first, iterator __before_last)
- void **splice_after** (iterator __pos, iterator __prev)
- void **splice_after** (iterator __pos, slist &__x)

- void **swap** (slist &__x)
- void **unique** ()
- template<class _BinaryPredicate>
  void **unique** (_BinaryPredicate __pred)

Static Private Member Functions

- static constexpr size_type max_size (const _Alloc &__a) noexcept

5.895.1 Detailed Description

template<class _Tp, class _Alloc = std::allocator<_Tp>>
class __gnu_cxx::slist< _Tp, _Alloc >

This is an SGI extension.

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

The documentation for this class was generated from the following file:

- slist

5.896 std::experimental::filesystem::v1::space_info Struct Reference

`#include <fs_fwd.h>`

Public Attributes

- uintmax_t **available**
- uintmax_t **capacity**
- uintmax_t **free**

5.896.1 Detailed Description

Information about free space on a disk.
The documentation for this struct was generated from the following file:

- experimental/bits/fs_fwd.h

5.897 std::filesystem::space_info Struct Reference

`#include <fs_fwd.h>`

Public Attributes

- uintmax_t **available**
- uintmax_t **capacity**
- uintmax_t **free**

Friends

- bool **operator==** (const space_info &, const space_info &)=default

5.897.1 Detailed Description

Information about free space on a disk.
The documentation for this struct was generated from the following file:

- bits/fs_fwd.h

5.898 __gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > Class Template Reference

`#include <splay_tree_.hpp>`

Public Types

- typedef _Alloc **allocator_type**
- typedef Cmp_Fn **cmp_fn**
- typedef std::pair< size_type, size_type > **comp_hash**
- typedef base_type::const_iterator **const_iterator**
- typedef base_type::const_pointer **const_pointer**
- typedef base_type::const_reference **const_reference**
- typedef base_type::const_reverse_iterator **const_reverse_iterator**
- typedef splay_tree_tag **container_category**
- typedef _Alloc::difference_type **difference_type**
- typedef base_type::iterator **iterator**
- typedef base_type::key_const_pointer **key_const_pointer**
- typedef base_type::key_const_reference **key_const_reference**
- typedef base_type::key_pointer **key_pointer**
- typedef base_type::key_reference **key_reference**
- typedef base_type::key_type **key_type**
- typedef base_type::mapped_const_pointer **mapped_const_pointer**
- typedef base_type::mapped_const_reference **mapped_const_reference**
- typedef base_type::mapped_pointer **mapped_pointer**
- typedef base_type::mapped_reference **mapped_reference**
- typedef base_type::mapped_type **mapped_type**
- typedef __nothrowcopy::indicator **no_throw_indicator**
- typedef traits_type::node_const_iterator **node_const_iterator**
- typedef traits_type::node_iterator **node_iterator**
- typedef base_type::node_update **node_update**
- typedef base_type::const_iterator **point_const_iterator**
- typedef base_type::point_iterator **point_iterator**
- typedef base_type::pointer **pointer**
- typedef base_type::reference **reference**
- typedef base_type::reverse_iterator **reverse_iterator**
- typedef _Alloc::size_type **size_type**
- typedef integral_constant< int, Store_Hash > **store_extra**
- typedef stored_data< value_type, size_type, Store_Hash > **stored_data_type**
- typedef base_type::value_type **value_type**

Public Member Functions

- **splay_tree_map** (const Cmp_Fn &)
- **splay_tree_map** (const Cmp_Fn &, const node_update &)
- **splay_tree_map** (const splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- template<typename It>
  void **copy_from_range** (It, It)
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- iterator **erase** (iterator it)
- bool **erase** (key_const_reference)
- reverse_iterator **erase** (reverse_iterator)
- template<typename Pred>
  size_type **erase_if** (Pred)
- point_iterator **find** (key_const_reference)
- point_const_iterator **find** (key_const_reference) const
- Cmp_Fn & **get_cmp_fn** ()
- const Cmp_Fn & **get_cmp_fn** () const
- void **initialize** ()
- std::pair< point_iterator, bool > **insert** (const_reference r_value)
- void **join** (splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- point_iterator **lower_bound** (key_const_reference)
- point_const_iterator **lower_bound** (key_const_reference) const
- size_type **max_size** () const
- node_iterator node_begin ()
- node_const_iterator node_begin () const
- node_iterator node_end ()
- node_const_iterator node_end () const
- mapped_reference **operator[ ]** (key_const_reference r_key)
- reverse_iterator **rbegin** ()
- const_reverse_iterator **rbegin** () const
- reverse_iterator **rend** ()
- const_reverse_iterator **rend** () const
- size_type **size** () const
- void **split** (key_const_reference, splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- void **swap** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- void **swap** (splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- point_iterator **upper_bound** (key_const_reference)
- point_const_iterator **upper_bound** (key_const_reference) const

Public Attributes

- no_throw_indicator **m_no_throw_copies_indicator**
- no_throw_indicator **m_no_throw_copies_indicator**
- store_extra **m_store_extra_indicator**
- store_extra **m_store_extra_indicator**

Protected Types

- typedef node_alloc_traits::value_type **node**
- typedef node_alloc_traits::allocator_type **node_allocator**
- typedef traits_type::null_node_update_pointer **null_node_update_pointer**
- typedef types_traits< Key, Mapped, _Alloc, false > **traits_base**

Protected Member Functions

- void **actual_erase_node** (node_pointer)
- template<typename Node_Update_>
  void **apply_update** (node_pointer, Node_Update_ *)
- void **apply_update** (node_pointer, null_node_update_pointer)
- std::pair< node_pointer, bool > **erase** (node_pointer)
- node_pointer **get_new_node_for_leaf_insert** (const_reference, false_type)
- node_pointer **get_new_node_for_leaf_insert** (const_reference, true_type)
- void **initialize_min_max** ()
- iterator **insert_imp_empty** (const_reference)
- std::pair< point_iterator, bool > **insert_leaf** (const_reference)
- iterator **insert_leaf_new** (const_reference, node_pointer, bool)
- void **join_finish** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- bool **join_prep** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- size_type **recursive_count** (node_pointer) const
- void **rotate_left** (node_pointer)
- void **rotate_parent** (node_pointer)
- void **rotate_right** (node_pointer)
- void **split_finish** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- bool **split_prep** (key_const_reference, bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _↩ Alloc > &)
- void **update_min_max_for_erased_node** (node_pointer)
- void **update_subtree_size** (node_pointer)
- template<typename Node_Update_>
  void **update_to_top** (node_pointer, Node_Update_ *)
- void **update_to_top** (node_pointer, null_node_update_pointer)
- void **value_swap** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)

Static Protected Member Functions

- static void **clear_imp** (node_pointer)

Protected Attributes

- node_pointer **m_p_head**
- size_type **m_size**

Static Protected Attributes

- static node_allocator **s_node_allocator**

5.898.1 Detailed Description

template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _↩ Alloc>
class __gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >

Splay tree.

5.898.2 Member Function Documentation

node_begin() [1/2]

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _↩
Alloc>
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator __gnu_↩
pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin
() [inline], [inherited]
```

Returns a node_iterator corresponding to the node at the root of the tree.

node_begin() [2/2]

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _↩
Alloc>
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __↩
gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_↩
begin () const [inline], [inherited]
```

Returns a const node_iterator corresponding to the node at the root of the tree.

node_end() [1/2]

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _↩
Alloc>
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator __gnu_↩
pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end ()
[inline], [inherited]
```

Returns a node_iterator corresponding to a node just after a leaf of the tree.

node_end() [2/2]

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _↩
Alloc>
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __↩
gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_↩
end () const [inline], [inherited]
```

Returns a const node_iterator corresponding to a node just after a leaf of the tree.

The documentation for this class was generated from the following file:

- splay_tree_.hpp

5.899 __gnu_pbds::detail::splay_tree_node_< Value_Type, Metadata, _Alloc > Struct Template Reference

```
#include <node.hpp>
```

Public Types

- typedef rebind_traits< _Alloc, metadata_type >::const_reference **metadata_const_reference**
- typedef rebind_traits< _Alloc, metadata_type >::reference **metadata_reference**
- typedef Metadata **metadata_type**
- typedef rebind_traits< _Alloc, splay_tree_node_ >::pointer **node_pointer**
- typedef rebind_traits< _Alloc, splay_tree_node_ >::size_type **size_type**
- typedef Value_Type **value_type**

Public Member Functions

- metadata_reference **get_metadata** ()
- metadata_const_reference **get_metadata** () const
- bool **special** () const

Public Attributes

- metadata_type **m_metadata**
- node_pointer **m_p_left**
- node_pointer **m_p_parent**
- node_pointer **m_p_right**
- bool **m_special**
- size_type **m_subtree_size**
- value_type **m_value**

5.899.1 Detailed Description

template<typename Value_Type, class Metadata, typename _Alloc>
struct __gnu_pbds::detail::splay_tree_node_< Value_Type, Metadata, _Alloc >

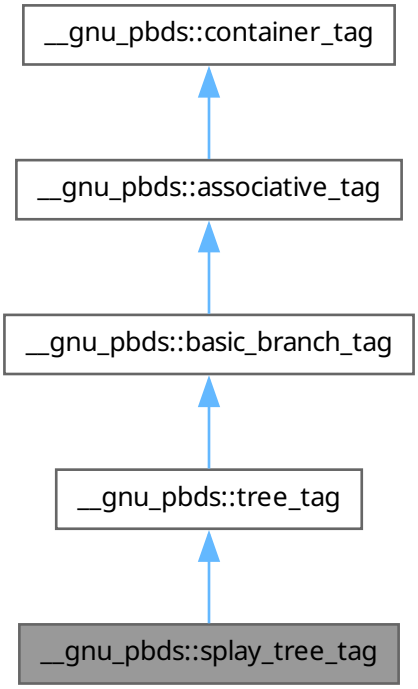
Node for splay tree.
The documentation for this struct was generated from the following file:

- splay_tree_/node.hpp

5.900 __gnu_pbds::splay_tree_tag Struct Reference

`#include <tag_and_trait.hpp>`

Inheritance diagram for __gnu_pbds::splay_tree_tag:



5.900.1 Detailed Description

Splay tree.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

5.901 std::stack< _Tp, _Sequence > Class Template Reference

`#include <stl_stack.h>`

Public Types

- typedef _Sequence::const_reference **const_reference**
- typedef _Sequence **container_type**
- typedef _Sequence::reference **reference**
- typedef _Sequence::size_type **size_type**
- typedef _Sequence::value_type **value_type**

Public Member Functions

- template<typename _Seq = _Sequence, typename _Requires = typename enable_if<is_default_constructible<_Seq>::value>::type> stack ()

- **stack** (_Sequence &&__c)
- template<typename _Alloc, typename _Requires = _Uses<_Alloc>>
  stack (_Sequence &&__c, const _Alloc &__a)
- template<typename _Alloc, typename _Requires = _Uses<_Alloc>>
  stack (const _Alloc &__a)
- **stack** (const _Sequence &__c)
- template<typename _Alloc, typename _Requires = _Uses<_Alloc>>
  stack (const _Sequence &__c, const _Alloc &__a)
- template<typename _Alloc, typename _Requires = _Uses<_Alloc>>
  stack (const stack &__q, const _Alloc &__a)
- template<typename _Alloc, typename _Requires = _Uses<_Alloc>>
  stack (stack &&__q, const _Alloc &__a)
- template<typename... _Args>
  decltype(auto) **emplace** (_Args &&... __args)
- bool empty () const
- void pop ()
- void push (const value_type &__x)
- void **push** (value_type &&__x)
- size_type size () const
- void **swap** (stack &__s) noexcept(__is_nothrow_swappable< _Sequence >::value)
- reference top ()
- const_reference top () const

Protected Attributes

- _Sequence **c**

Friends

- template<typename _Tp1, typename _Seq1>
  bool **operator**< (const stack< _Tp1, _Seq1 > &, const stack< _Tp1, _Seq1 > &)
- template<typename _Tp1, three_way_comparable _Seq1>
  compare_three_way_result_t< _Seq1 > **operator**<=> (const stack< _Tp1, _Seq1 > &, const stack< _Tp1, _Seq1 > &)
- template<typename _Tp1, typename _Seq1>
  bool **operator==** (const stack< _Tp1, _Seq1 > &, const stack< _Tp1, _Seq1 > &)

5.901.1 Detailed Description

template<typename _Tp, typename _Sequence = deque<_Tp>>
class std::stack< _Tp, _Sequence >

A standard container giving FILO behavior.

Template Parameters

_Tp	Type of element.
_Sequence	Type of underlying sequence, defaults to deque<_Tp>.

Meets many of the requirements of a `container`, but does not define anything to do with iterators. Very few of the other standard container interfaces are defined.
This is not a true container, but an *adaptor*. It holds another container, and provides a wrapper interface to that container. The wrapper is what enforces strict first-in-last-out stack behavior.

The second template parameter defines the type of the underlying sequence/container. It defaults to std::deque, but it can be any type that supports `back`, `push_back`, and `pop_back`, such as std::list, std::vector, or an appropriate user-defined type.
Members not found in *normal* containers are `container_type`, which is a typedef for the second Sequence parameter, and `push`, `pop`, and `top`, which are standard stack/FILO operations.

5.901.2 Constructor & Destructor Documentation

stack()

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
template<typename _Seq = _Sequence, typename _Requires = typename enable_if<is_default_constructible<↩
_Seq>::value>::type>
std::stack< _Tp, _Sequence >::stack ()  [inline]
```

Default constructor creates no elements.

5.901.3 Member Function Documentation

empty()

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
bool std::stack< _Tp, _Sequence >::empty () const  [inline], [nodiscard]
```

Returns true if the stack is empty.

pop()

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
void std::stack< _Tp, _Sequence >::pop ()  [inline]
```

Removes first element.
This is a typical stack operation. It shrinks the stack by one. The time complexity of the operation depends on the underlying sequence.
Note that no data is returned, and if the first element's data is needed, it should be retrieved before pop() is called.

push()

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
void std::stack< _Tp, _Sequence >::push (
            const value_type & __x)  [inline]
```

Add data to the top of the stack.

Parameters

_↩ _x	Data to be added.

This is a typical stack operation. The function creates an element at the top of the stack and assigns the given data to it. The time complexity of the operation depends on the underlying sequence.

size()

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
size_type std::stack< _Tp, _Sequence >::size () const  [inline], [nodiscard]
```

Returns the number of elements in the stack.

top() [1/2]

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
reference std::stack< _Tp, _Sequence >::top () [inline], [nodiscard]
```

Returns a read/write reference to the data at the first element of the stack.

top() [2/2]

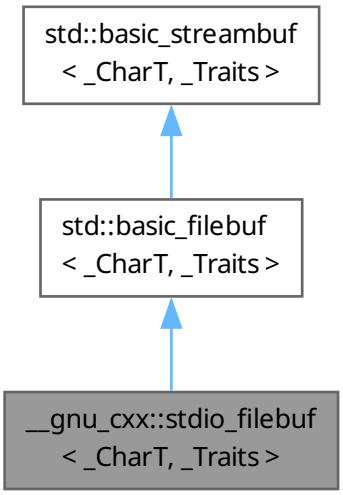
```
template<typename _Tp, typename _Sequence = deque<_Tp>>
const_reference std::stack< _Tp, _Sequence >::top () const [inline], [nodiscard]
```

Returns a read-only (constant) reference to the data at the first element of the stack.
The documentation for this class was generated from the following file:

- stl_stack.h

5.902 __gnu_cxx::stdio_filebuf< _CharT, _Traits > Class Template Reference

`#include <stdio_filebuf.h>`
Inheritance diagram for __gnu_cxx::stdio_filebuf< _CharT, _Traits >:



Public Types

- typedef codecvt< char_type, char, __state_type > **__codecvt_type**
- typedef __basic_file< char > **__file_type**
- typedef basic_filebuf< char_type, traits_type > **__filebuf_type**
- typedef traits_type::state_type **__state_type**
- typedef basic_streambuf< char_type, traits_type > **__streambuf_type**
- typedef _CharT **char_type**
- typedef traits_type::int_type **int_type**
- typedef traits_type::off_type **off_type**
- typedef traits_type::pos_type **pos_type**

- typedef std::size_t **size_t**
- typedef _Traits **traits_type**

Public Member Functions

- stdio_filebuf ()
- stdio_filebuf (int __fd, std::ios_base::openmode __mode, size_t __size=static_cast< size_t >(BUFSIZ))
- stdio_filebuf (std::__c_file *__f, std::ios_base::openmode __mode, size_t __size=static_cast< size_t >(BUFSIZ))
- **stdio_filebuf** (stdio_filebuf &&)=default
- virtual ~stdio_filebuf ()
- __filebuf_type * close ()
- int fd ()
- std::__c_file * file ()
- locale getloc () const
- streamsize in_avail ()
- bool is_open () const throw ()
- template<typename _Path>
  _If_fs_path< _Path, __filebuf_type * > open (const _Path &__s, ios_base::openmode __mode)
- __filebuf_type * open (const char *__s, ios_base::openmode __mode)
- __filebuf_type * open (const std::string &__s, ios_base::openmode __mode)
- stdio_filebuf & **operator=** (stdio_filebuf &&)=default
- locale pubimbue (const locale &__loc)
- int_type sbumpc ()
- int_type sgetc ()
- streamsize sgetn (char_type *__s, streamsize __n)
- int_type snextc ()
- int_type sputbackc (char_type __c)
- int_type sputc (char_type __c)
- streamsize sputn (const char_type *__s, streamsize __n)
- int_type sungetc ()
- void **swap** (stdio_filebuf &__fb)
- void **swap** (basic_filebuf &)

- basic_streambuf * pubsetbuf (char_type *__s, streamsize __n)
- pos_type pubseekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)
- pos_type pubseekpos (pos_type __sp, ios_base::openmode __mode=ios_base::in|ios_base::out)
- int pubsync ()

Protected Member Functions

- void **__safe_gbump** (streamsize __n)
- void **__safe_pbump** (streamsize __n)
- void **_M_allocate_internal_buffer** ()
- bool **_M_convert_to_external** (char_type *, streamsize)
- void _M_create_pback ()
- void **_M_destroy_internal_buffer** () throw ()
- void _M_destroy_pback () throw ()
- int **_M_get_ext_pos** (__state_type &__state)
- pos_type **_M_seek** (off_type __off, ios_base::seekdir __way, __state_type __state)
- void _M_set_buffer (streamsize __off)

- bool **_M_terminate_output** ()
- void gbump (int __n)
- virtual void imbue (const locale &__loc)
- virtual int_type overflow (int_type __c=_Traits::eof())
- virtual int_type pbackfail (int_type __c=_Traits::eof())
- void pbump (int __n)
- virtual pos_type seekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base↩ ::in|ios_base::out)
- virtual pos_type seekpos (pos_type __pos, ios_base::openmode __mode=ios_base::in|ios_base::out)
- virtual __streambuf_type * setbuf (char_type *__s, streamsize __n)
- void setg (char_type *__gbeg, char_type *__gnext, char_type *__gend)
- void setp (char_type *__pbeg, char_type *__pend)
- virtual streamsize showmanyc ()
- void **swap** (basic_streambuf &__sb)
- virtual int sync ()
- virtual int_type uflow ()
- virtual int_type underflow ()
- virtual streamsize xsgetn (char_type *__s, streamsize __n)
- virtual streamsize xsputn (const char_type *__s, streamsize __n)

- char_type * eback () const
- char_type * gptr () const
- char_type * egptr () const

- char_type * pbase () const
- char_type * pptr () const
- char_type * epptr () const

Protected Attributes

- char_type * _M_buf
- bool **_M_buf_allocated**
- locale _M_buf_locale
- size_t _M_buf_size
- const __codecvt_type * **_M_codecvt**
- char * _M_ext_buf
- streamsize _M_ext_buf_size
- char * **_M_ext_end**
- const char * _M_ext_next
- __file_type **_M_file**
- char_type * _M_in_beg
- char_type * _M_in_cur
- char_type * _M_in_end
- __c_lock **_M_lock**
- ios_base::openmode _M_mode
- char_type * _M_out_beg
- char_type * _M_out_cur
- char_type * _M_out_end
- bool _M_reading

- __state_type **_M_state_beg**
- __state_type **_M_state_cur**
- __state_type **_M_state_last**
- bool **_M_writing**

- char_type _M_pback
- char_type * _M_pback_cur_save
- char_type * _M_pback_end_save
- bool _M_pback_init

5.902.1 Detailed Description

template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
class __gnu_cxx::stdio_filebuf< _CharT, _Traits >

Provides a layer of compatibility for C/POSIX.
This GNU extension provides extensions for working with standard C FILE*'s and POSIX file descriptors. It must be instantiated by the user with the type of character used in the file stream, e.g., stdio_filebuf<char>.

5.902.2 Constructor & Destructor Documentation

stdio_filebuf() [1/3]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf ()  [inline]
```

deferred initialization
References std::basic_filebuf< _CharT, _Traits >::basic_filebuf().

stdio_filebuf() [2/3]

```
template<typename _CharT, typename _Traits>
__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf (
            int __fd,
            std::ios_base::openmode __mode,
            size_t __size = static_cast<size_t>( BUFSIZ ))
```

Parameters

__fd	An open file descriptor.
__mode	Same meaning as in a standard filebuf.
__size	Optimal or preferred size of internal buffer, in chars.

This constructor associates a file stream buffer with an open POSIX file descriptor. The file descriptor will be automatically closed when the stdio_filebuf is closed/destroyed.
References std::basic_filebuf< _CharT, _Traits >::_M_buf_size, std::basic_filebuf< _CharT, _Traits >::_M_mode, std::basic_filebuf< _CharT, _Traits >::_M_reading, std::basic_filebuf< _CharT, _Traits >::_M_set_buffer(), and std::basic_filebuf< _CharT, _Traits >::is_open().

stdio_filebuf() [3/3]

```
template<typename _CharT, typename _Traits>
__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf (
            std::__c_file * __f,
            std::ios_base::openmode __mode,
            size_t __size = static_cast<size_t>( BUFSIZ ))
```

Parameters

__f	An open FILE*.
__mode	Same meaning as in a standard filebuf.
__size	Optimal or preferred size of internal buffer, in chars. Defaults to system's BUFSIZ.

This constructor associates a file stream buffer with an open C FILE*. The FILE* will not be automatically closed when the stdio_filebuf is closed/destroyed.
References std::basic_filebuf< _CharT, _Traits >::_M_buf_size, std::basic_filebuf< _CharT, _Traits >::_M_mode, std::basic_filebuf< _CharT, _Traits >::_M_reading, std::basic_filebuf< _CharT, _Traits >::_M_set_buffer(), and std::basic_filebuf< _CharT, _Traits >::is_open().

~stdio_filebuf()

```
template<typename _CharT, typename _Traits>
__gnu_cxx::stdio_filebuf< _CharT, _Traits >::~stdio_filebuf ()  [virtual]
```
Closes the external data stream if the file descriptor constructor was used.

5.902.3 Member Function Documentation

_M_create_pback()

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf< _CharT, _Traits >::_M_create_pback ()  [inline], [protected], [inherited]
```
Initializes pback buffers, and moves normal buffers to safety. Assumptions: _M_in_cur has already been moved back
Referenced by pbackfail().

_M_destroy_pback()

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf< _CharT, _Traits >::_M_destroy_pback () throw ( )  [inline], [protected],
[inherited]
```
Deactivates pback buffer contents, and restores normal buffer. Assumptions: The pback buffer has only moved forward.
Referenced by overflow(), seekoff(), seekpos(), underflow(), and xsgetn().

_M_set_buffer()

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf< _CharT, _Traits >::_M_set_buffer (
             streamsize __off)  [inline], [protected], [inherited]
```
This function sets the pointers of the internal buffer, both get and put areas. Typically:
__off == egptr() - eback() upon underflow/uflow (**read** mode); __off == 0 upon overflow (**write** mode); __off == -1 upon open, setbuf, seekoff/pos (**uncommitted** mode).
NB: epptr() - pbase() == _M_buf_size - 1, since _M_buf_size reflects the actual allocated memory and the last cell is reserved for the overflow char of a full put area.
Referenced by __gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf(), __gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf(), close(), imbue(), open(), overflow(), pbackfail(), underflow(), xsgetn(), and xsputn().

close()

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::__filebuf_type * std::basic_filebuf< _CharT, _Traits >::close
()  [inherited]
```
Closes the currently associated file.

Returns

`this` on success, NULL on failure

If no file is currently open, this function immediately fails.
If a *put buffer area* exists, `overflow(eof)` is called to flush all the characters. The file is then closed.
If any operations fail, this function also fails.
References basic_filebuf(), _M_mode, _M_pback_init, _M_reading, _M_set_buffer(), and is_open().
Referenced by open().

eback()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::eback () const  [inline], [protected], [inherited]
```

Access to the get area.
These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_stringbuf< _Cha std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_filebuf< _CharT, _Traits >::underflow(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

egptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::egptr () const  [inline], [protected], [inherited]
```

Access to the get area.
These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Referenced by std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_filebuf< _CharT, _Traits >::underflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), std::wbuffer_convert< _Codecvt, _Elem, _Tr >::underflow(), std::basic_filebuf< _CharT, _Traits >::xsgetn(), and xsgetn().

epptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::epptr () const  [inline], [protected], [inherited]
```

Access to the put area.
These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_filebuf< _CharT, _Traits >::xsputn(), and xsputn().

fd()

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
int __gnu_cxx::stdio_filebuf< _CharT, _Traits >::fd ()  [inline]
```

Returns

The underlying file descriptor.

Once associated with an external data stream, this function can be used to access the underlying POSIX file descriptor. Note that there is no way for the library to track what you do with the descriptor, so be careful.

file()

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
std::__c_file * __gnu_cxx::stdio_filebuf< _CharT, _Traits >::file ()  [inline]
```

Returns

The underlying FILE*.

This function can be used to access the underlying "C" file pointer. Note that there is no way for the library to track what you do with the file, so be careful.

gbump()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::gbump (
            int __n)  [inline], [protected], [inherited]
```

Moving the read position.

Parameters

_↩ _n	The delta by which to move.

This just advances the read position without returning any data.
Referenced by std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

getloc()

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::getloc () const  [inline], [inherited]
```

Locale access.

Returns

The current locale in effect.

If pubimbue(loc) has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

gptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::gptr () const  [inline], [protected], [inherited]
```

Access to the get area.
These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_stringbuf< _Cha std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_filebuf< _CharT, _Traits >::underflow(), std::basic_stringbuf< _CharT, _T std::wbuffer_convert< _Codecvt, _Elem, _Tr >::underflow(), std::basic_filebuf< _CharT, _Traits >::xsgetn(), and xsgetn().

imbue()

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf< _CharT, _Traits >::imbue (
            const locale & __loc)  [protected], [virtual], [inherited]
```

Changes translations.

Parameters

__loc	A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

Note

Base class version does nothing.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_ext_buf, _M_ext_next, _M_mode, _M_reading, _M_set_buffer(), std::ios_base::cur, std::basic_streambuf< _CharT, _Trait std::basic_streambuf< _CharT, _Traits >::gptr(), is_open(), and seekoff().

in_avail()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::in_avail ()  [inline], [inherited]
```

Looking ahead into the stream.

Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

is_open()

```
template<typename _CharT, typename _Traits>
bool std::basic_filebuf< _CharT, _Traits >::is_open () const throw ( )   [inline], [nodiscard],
[inherited]
```

Returns true if the external file is open.
Referenced by __gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf(), __gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf(), close(), imbue(), open(), seekoff(), seekpos(), setbuf(), and showmanyc().

open() [1/3]

```
template<typename _CharT, typename _Traits>
template<typename _Path>
_If_fs_path< _Path, __filebuf_type * > std::basic_filebuf< _CharT, _Traits >::open (
            const _Path & __s,
            ios_base::openmode __mode)  [inline], [inherited]
```

Opens an external file.

Parameters

__s	The name of the file, as a filesystem::path.
__mode	The open mode flags.

Returns

`this` on success, NULL on failure

open() [2/3]

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::__filebuf_type * std::basic_filebuf< _CharT, _Traits >::open (
            const char * __s,
            ios_base::openmode __mode)  [inherited]
```

Opens an external file.

Parameters

__s	The name of the file.
__mode	The open mode flags.

Returns

`this` on success, NULL on failure

If a file is already open, this function immediately fails. Otherwise it tries to open the file named *__s* using the flags given in *__mode*.

Table 92, adapted here, gives the relation between openmode combinations and the equivalent `fopen()` flags. (NB: lines app, in|out|app, in|app, binary|app, binary|in|out|app, and binary|in|app per DR 596)

ios_base Flag combination				stdio equivalent
binary	in	out	trunc	app	
		+			w
		+		+	a
				+	a
		+	+		w
	+				r
	+	+			r+
	+	+	+		w+
	+	+		+	a+
	+			+	a+
	+		+		wb

```
|    +          +            +             ab                  |
|    +                       +             ab                  |
|    +          +      +                   wb                  |
|    +      +                               rb                  |
|    +      +    +                          r+b                 |
|    +      +    +      +                   w+b                 |
|    +      +    +            +             a+b                 |
|    +      +                 +             a+b                 |
+-------------------------------------------------------------+
```

References _M_mode, _M_reading, _M_set_buffer(), std::ios_base::ate, close(), std::ios_base::end, is_open(), and seekoff().

open() [3/3]

```
template<typename _CharT, typename _Traits>
__filebuf_type * std::basic_filebuf< _CharT, _Traits >::open (
            const std::string & __s,
            ios_base::openmode __mode)  [inline], [inherited]
```

Opens an external file.

Parameters

__s	The name of the file.
__mode	The open mode flags.

Returns

`this` on success, NULL on failure

overflow()

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::int_type std::basic_filebuf< _CharT, _Traits >::overflow (
            int_type __c = _Traits::eof())  [protected], [virtual], [inherited]
```

Consumes data from the buffer; writes to the controlled sequence.

Parameters

__c	An additional character to consume.

Returns

eof() to indicate failure, something else (usually __*c*, or not_eof())

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if __*c* is not `eof()`.
For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.
A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_buf_size, _M_destroy_pback(), _M_mode, _M_reading, _M_set_buffer(), std::ios_base::app, std::ios_base::cur, std::ios_base::out, std::basic_streambuf< _CharT, _Traits >::pbase(), std::basic_streambuf< _CharT, _Traits >::pbum
and std::basic_streambuf< _CharT, _Traits >::pptr().
Referenced by pbackfail(), sync(), underflow(), and xsgetn().

pbackfail()

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::int_type std::basic_filebuf< _CharT, _Traits >::pbackfail (
            int_type __c = _Traits::eof())  [protected], [virtual], [inherited]
```

Tries to back up the input sequence.

Parameters

_↩ _c	The character to be inserted back into the sequence.

Returns

eof() on failure, *some other value* on success

Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_create_pback(), _M_mode, _M_pback_init, _M_reading, _M_set_buffer(), std::ios_base::cur, std::basic_streambuf< _CharT, _Traits >::eback(), std::basic_streambuf< _CharT, _Traits >::gbump(), std::basic_streambuf< _CharT, _
std::ios_base::in, overflow(), seekoff(), and underflow().

pbase()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::pbase () const  [inline], [protected], [inherited]
```

Access to the put area.
These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, std::basic_filebuf< _CharT, _Traits >::sync(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

pbump()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::pbump (
            int __n)  [inline], [protected], [inherited]
```

Moving the write position.

Parameters

_↩ _n	The delta by which to move.

This just advances the write position without returning any data.
Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow().

pptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::pptr () const  [inline], [protected], [inherited]
```

Access to the put area.
These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_filebuf< _CharT, _T std::basic_filebuf< _CharT, _Traits >::xsputn(), and xsputn().

pubimbue()

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::pubimbue (
            const locale & __loc)  [inline], [inherited]
```

Entry point for imbue().

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls the derived imbue(__loc).

pubseekoff()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekoff (
            off_type __off,
            ios_base::seekdir __way,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [inline], [inherited]
```

Alters the stream position.

Parameters

__off	Offset.
__way	Value for ios_base::seekdir.
__mode	Value for ios_base::openmode.

Calls virtual seekoff function.

pubseekpos()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekpos (
            pos_type __sp,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [inline], [inherited]
```

Alters the stream position.

Parameters

__sp	Position
__mode	Value for ios_base::openmode.

Calls virtual seekpos function.

pubsetbuf()

```
template<typename _CharT, typename _Traits>
basic_streambuf * std::basic_streambuf< _CharT, _Traits >::pubsetbuf (
            char_type * __s,
            streamsize __n)  [inline], [inherited]
```

Entry points for derived buffer functions.
The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

pubsync()

```
template<typename _CharT, typename _Traits>
int std::basic_streambuf< _CharT, _Traits >::pubsync ()  [inline], [inherited]
```

Calls virtual sync function.
Referenced by std::basic_istream< _CharT, _Traits >::sync().

sbumpc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sbumpc ()  [inline], [inherited]
```

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.
Referenced by std::basic_istream< _CharT, _Traits >::getline().

seekoff()

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::pos_type std::basic_filebuf< _CharT, _Traits >::seekoff (
            off_type ,
            ios_base::seekdir ,
            ios_base::openmode  = ios_base::in | ios_base::out)  [protected], [virtual], [inherited]
```

Alters the stream positions.
Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_destroy_pback(), _M_reading, std::ios_base::cur, is_open(), std::basic_streambuf< _CharT, _Traits >::pbase(), and std::basic_streambuf< _CharT, _Traits >::pptr().
Referenced by imbue(), open(), and pbackfail().

seekpos()

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::pos_type std::basic_filebuf< _CharT, _Traits >::seekpos (
            pos_type ,
            ios_base::openmode  = ios_base::in | ios_base::out)  [protected], [virtual], [inherited]
```

Alters the stream positions.
Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_destroy_pback(), std::ios_base::beg, and is_open().

setbuf()

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::__streambuf_type * std::basic_filebuf< _CharT, _Traits >↩
::setbuf (
            char_type * __s,
            streamsize __n)  [protected], [virtual], [inherited]
```

Manipulates the buffer.

Parameters

_↩ _s	Pointer to a buffer area.
_↩ _n	Size of __s.

Returns

`this`

If no file has been opened, and both __s and __n are zero, then the stream becomes unbuffered. Otherwise, __s is used as a buffer; see https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.↩ html#io.streambuf.buffering for more.
Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_buf, _M_buf_size, and is_open().

setg()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setg (
            char_type * __gbeg,
            char_type * __gnext,
            char_type * __gend)  [inline], [protected], [inherited]
```

Setting the three read area pointers.

Parameters

__gbeg	A pointer.
__gnext	A pointer.
__gend	A pointer.

Postcondition

__gbeg == `eback()`, *__gnext* == `gptr()`, and *__gend* == `egptr()`

Referenced by std::wbuffer_convert< _Codecvt, _Elem, _Tr >::wbuffer_convert(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflo
std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(),
and std::basic_filebuf< _CharT, _Traits >::xsgetn().

setp()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setp (
            char_type * __pbeg,
            char_type * __pend)  [inline], [protected], [inherited]
```

Setting the three write area pointers.

Parameters

__pbeg	A pointer.
__pend	A pointer.

Postcondition

__pbeg == `pbase()`, *__pbeg* == `pptr()`, and *__pend* == `epptr()`

Referenced by std::wbuffer_convert< _Codecvt, _Elem, _Tr >::wbuffer_convert().

sgetc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sgetc ()  [inline], [inherited]
```

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.
Referenced by std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(),
std::basic_istream< _CharT, _Traits >::ignore(), and std::basic_istream< _CharT, _Traits >::ignore().

sgetn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sgetn (
            char_type * __s,
            streamsize __n)  [inline], [inherited]
```

Entry point for xsgetn.

Parameters

_↩ _s	A buffer area.
_↩ _n	A count.

Returns xsgetn(__s,__n). The effect is to fill __s[0] through __s[__n-1] with characters from the input sequence, if possible.

showmanyc()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_filebuf< _CharT, _Traits >::showmanyc ()  [protected], [virtual], [inherited]
```

Investigating the data available.

Returns

An estimate of the number of characters available in the input sequence, or -1.

If it returns a positive value, then successive calls to `underflow()` *will not return* `traits::eof()` *until at least that number of characters have been supplied. If* `showmanyc()` *returns -1, then calls to* `underflow()` *or* `uflow()` *will fail.* [27.5.2.4.3]/1

Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to underflow or uflow] will not return* `eof()` *but that they will return immediately.*

The standard adds that *the morphemes of* `showmanyc` *are* ***es-how-many-see****, not* ***show-manic****.*

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_mode, std::ios_base::binary, std::basic_streambuf< _CharT, _Traits >::egptr(), std::basic_streambuf< _CharT, _Traits > std::ios_base::in, and is_open().

snextc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::snextc ()  [inline], [inherited]
```

Getting the next character.

Returns

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.
Referenced by std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< char_type, traits_type >::ignore(), std::basic_istream< char_type, traits_type >::operator>>(), and std::basic_istream< char_type, traits_type >::putback().

sputbackc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputbackc (
            char_type __c)  [inline], [inherited]
```

Pushing characters back into the input stream.

Parameters

_↩ _c	The character to push back.

Returns

The previous character, if possible.

Similar to sungetc(), but __c is pushed onto the stream instead of *the previous character.* If successful, the next character fetched from the input stream will be __c.
Referenced by std::basic_istream< _CharT, _Traits >::putback().

sputc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputc (
            char_type __c)  [inline], [inherited]
```

Entry point for all single-character output functions.

Parameters

_↩ _c	A character to output.

Returns

__c, if possible.

One of two public output functions.
If a write position is available for the output sequence (i.e., the buffer is not full), stores __c in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(↩ __c)`.
Referenced by std::wbuffer_convert< _Codecvt, _Elem, _Tr >::overflow().

sputn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sputn (
            const char_type * __s,
            streamsize __n)  [inline], [inherited]
```

Entry point for all single-character output functions.

Parameters

_↩ _s	A buffer read area.
_↩ _n	A count.

One of two public output functions.
Returns xsputn(__s,__n). The effect is to write __*s*[0] through __*s*[__n-1] to the output sequence, if possible.

sungetc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sungetc ()  [inline], [inherited]
```

Moving backwards in the input stream.

Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns pbackfail(). The effect is to *unget* the last character *gotten*.
Referenced by std::basic_istream< char_type, traits_type >::sentry::sentry().

sync()

```
template<typename _CharT, typename _Traits>
int std::basic_filebuf< _CharT, _Traits >::sync ()  [protected], [virtual], [inherited]
```

Synchronizes the buffer arrays with the controlled sequences.

Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

Note

Base class version does nothing, returns zero.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References overflow(), std::basic_streambuf< _CharT, _Traits >::pbase(), and std::basic_streambuf< _CharT, _Traits >::pptr().

uflow()

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::uflow ()  [inline], [protected], [virtual],
[inherited]
```

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.
Reimplemented in __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >.
Referenced by xsgetn().

underflow()

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::int_type std::basic_filebuf< _CharT, _Traits >::underflow ()
[protected], [virtual], [inherited]
```

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.
For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.
A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_buf_size, _M_destroy_pback(), _M_mode, _M_set_buffer(), std::basic_streambuf< _CharT, _Traits >::eback(), std::basic_streambuf< _CharT, _Traits >::egptr(), std::basic_streambuf< _CharT, _Traits >::gptr(), std::ios_base::in, and overflow().
Referenced by pbackfail().

xsgetn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_filebuf< _CharT, _Traits >::xsgetn (
             char_type * __s,
             streamsize __n)  [protected], [virtual], [inherited]
```

Multiple character extraction.

Parameters

_↩ _s	A buffer area.
_↩ _n	Maximum number of characters to assign.

Returns

The number of characters assigned.

Fills __s[0] through __s[__n-1] with characters from the input sequence, as if by `sbumpc()`. Stops when either __n characters have been copied, or when `traits::eof()` would be copied.
It is expected that derived classes provide a more efficient implementation by overriding this definition.
Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_buf_size, _M_destroy_pback(), _M_mode, _M_pback_init, _M_reading, _M_set_buffer(), std::basic_streambuf< _CharT, std::basic_streambuf< _CharT, _Traits >::egptr(), std::basic_streambuf< _CharT, _Traits >::gbump(), std::basic_streambuf< _CharT, _T std::ios_base::in, overflow(), std::basic_streambuf< _CharT, _Traits >::setg(), and std::basic_streambuf< char_type, traits_type >::xsgetn

xsputn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_filebuf< _CharT, _Traits >::xsputn (
            const char_type * __s,
            streamsize __n)  [protected], [virtual], [inherited]
```

Multiple character insertion.

Parameters

_↩ _s	A buffer area.
_↩ _n	Maximum number of characters to write.

Returns

The number of characters written.

Writes __s[0] through __s[__n-1] to the output sequence, as if by `sputc()`. Stops when either *n* characters have been copied, or when `sputc()` would return `traits::eof()`.
It is expected that derived classes provide a more efficient implementation by overriding this definition.
Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_buf_size, _M_mode, _M_reading, _M_set_buffer(), std::ios_base::app, std::basic_streambuf< _CharT, _Traits >::epptr(), std::ios_base::out, std::basic_streambuf< _CharT, _Traits >::pbase(), std::basic_streambuf< _CharT, _Traits >::pptr(), and std::basic_streambuf< char_type, traits_type >::xsputn().

5.902.4 Member Data Documentation

_M_buf

```
template<typename _CharT, typename _Traits>
char_type* std::basic_filebuf< _CharT, _Traits >::_M_buf  [protected], [inherited]
```

Pointer to the beginning of internal buffer.
Referenced by basic_filebuf(), and setbuf().

_M_buf_locale

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::_M_buf_locale  [protected], [inherited]
```

Current locale setting.
Referenced by std::basic_filebuf< _CharT, _Traits >::basic_filebuf().

_M_buf_size

```
template<typename _CharT, typename _Traits>
size_t std::basic_filebuf< _CharT, _Traits >::_M_buf_size  [protected], [inherited]
```

Actual size of internal buffer. This number is equal to the size of the put area + 1 position, reserved for the overflow char of a full area.
Referenced by basic_filebuf(), __gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf(), __gnu_cxx::stdio_filebuf< _CharT, _Traits >::s overflow(), setbuf(), underflow(), xsgetn(), and xsputn().

_M_ext_buf

```
template<typename _CharT, typename _Traits>
char* std::basic_filebuf< _CharT, _Traits >::_M_ext_buf  [protected], [inherited]
```

Buffer for external characters. Used for input when codecvt::always_noconv() == false. When valid, this corresponds to eback().
Referenced by basic_filebuf(), and imbue().

_M_ext_buf_size

```
template<typename _CharT, typename _Traits>
streamsize std::basic_filebuf< _CharT, _Traits >::_M_ext_buf_size  [protected], [inherited]
```

Size of buffer held by _M_ext_buf.
Referenced by basic_filebuf().

_M_ext_next

```
template<typename _CharT, typename _Traits>
const char* std::basic_filebuf< _CharT, _Traits >::_M_ext_next  [protected], [inherited]
```

Pointers into the buffer held by _M_ext_buf that delimit a subsequence of bytes that have been read but not yet converted. When valid, _M_ext_next corresponds to egptr().
Referenced by basic_filebuf(), and imbue().

_M_in_beg

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_beg  [protected], [inherited]
```

Start of get area.

_M_in_cur

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_cur  [protected], [inherited]
```

Current read area.

_M_in_end

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_end  [protected], [inherited]
```

End of get area.

_M_mode

```
template<typename _CharT, typename _Traits>
ios_base::openmode std::basic_filebuf< _CharT, _Traits >::_M_mode  [protected], [inherited]
```

Place to stash in || out || in | out settings for current filebuf.
Referenced by basic_filebuf(), __gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf(), __gnu_cxx::stdio_filebuf< _CharT, _Traits >::
close(), imbue(), open(), overflow(), pbackfail(), showmanyc(), underflow(), xsgetn(), and xsputn().

_M_out_beg

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_beg  [protected], [inherited]
```

Start of put area.

_M_out_cur

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_cur  [protected], [inherited]
```

Current put area.

_M_out_end

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_end  [protected], [inherited]
```
End of put area.

_M_pback

```
template<typename _CharT, typename _Traits>
char_type std::basic_filebuf< _CharT, _Traits >::_M_pback  [protected], [inherited]
```
Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Referenced by basic_filebuf().

_M_pback_cur_save

```
template<typename _CharT, typename _Traits>
char_type* std::basic_filebuf< _CharT, _Traits >::_M_pback_cur_save  [protected], [inherited]
```
Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Referenced by basic_filebuf().

_M_pback_end_save

```
template<typename _CharT, typename _Traits>
char_type* std::basic_filebuf< _CharT, _Traits >::_M_pback_end_save  [protected], [inherited]
```
Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Referenced by basic_filebuf().

_M_pback_init

```
template<typename _CharT, typename _Traits>
bool std::basic_filebuf< _CharT, _Traits >::_M_pback_init  [protected], [inherited]
```
Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Referenced by basic_filebuf(), close(), pbackfail(), and xsgetn().

_M_reading

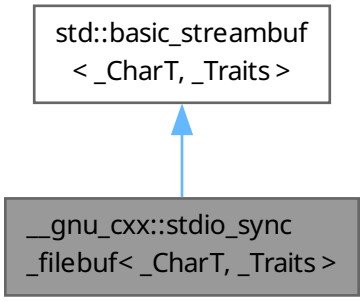
```
template<typename _CharT, typename _Traits>
bool std::basic_filebuf< _CharT, _Traits >::_M_reading  [protected], [inherited]
```
_M_reading == false && _M_writing == false for **uncommitted** mode; _M_reading == true for **read** mode; _M_writing == true for **write** mode;

NB: _M_reading == true && _M_writing == true is unused.
Referenced by basic_filebuf(), __gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf(), __gnu_cxx::stdio_filebuf< _CharT, _Traits >::
close(), imbue(), open(), overflow(), pbackfail(), seekoff(), xsgetn(), and xsputn().
The documentation for this class was generated from the following file:

- stdio_filebuf.h

5.903 __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits > Class Template Reference

`#include <stdio_sync_filebuf.h>`
Inheritance diagram for __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >:



Public Types

- typedef _CharT **char_type**
- typedef traits_type::int_type **int_type**
- typedef traits_type::off_type **off_type**
- typedef traits_type::pos_type **pos_type**
- typedef _Traits **traits_type**

Public Member Functions

- **stdio_sync_filebuf** (std::__c_file *__f)
- **stdio_sync_filebuf** (stdio_sync_filebuf &&__fb) noexcept
- std::__c_file * file ()
- locale getloc () const
- streamsize in_avail ()
- stdio_sync_filebuf & **operator=** (stdio_sync_filebuf &&__fb) noexcept
- locale pubimbue (const locale &__loc)
- int_type sbumpc ()
- int_type sgetc ()
- streamsize sgetn (char_type *__s, streamsize __n)
- int_type snextc ()
- int_type sputbackc (char_type __c)
- int_type sputc (char_type __c)
- streamsize sputn (const char_type *__s, streamsize __n)

- int_type sungetc ()
- void **swap** (stdio_sync_filebuf &__fb)

- basic_streambuf * pubsetbuf (char_type *__s, streamsize __n)
- pos_type pubseekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)
- pos_type pubseekpos (pos_type __sp, ios_base::openmode __mode=ios_base::in|ios_base::out)
- int pubsync ()

Protected Member Functions

- void **__safe_gbump** (streamsize __n)
- void **__safe_pbump** (streamsize __n)
- void gbump (int __n)
- virtual void imbue (const locale &__loc)
- virtual int_type overflow (int_type __c=traits_type::eof())
- virtual int_type pbackfail (int_type __c=traits_type::eof())
- void pbump (int __n)
- virtual std::streampos **seekoff** (std::streamoff __off, std::ios_base::seekdir __dir, std::ios_base::openmode=std::ios_base::in|std::ios
- virtual pos_type seekoff (off_type, ios_base::seekdir, ios_base::openmode=ios_base::in|ios_base::out)
- virtual std::streampos **seekpos** (std::streampos __pos, std::ios_base::openmode __mode=std::ios_base::in|std::ios_base::out)
- virtual pos_type seekpos (pos_type, ios_base::openmode=ios_base::in|ios_base::out)
- virtual basic_streambuf< char_type, _Traits > * setbuf (char_type *, streamsize)
- void setg (char_type *__gbeg, char_type *__gnext, char_type *__gend)
- void setp (char_type *__pbeg, char_type *__pend)
- virtual streamsize showmanyc ()
- void **swap** (basic_streambuf &__sb)
- virtual int sync ()
- int_type **syncgetc** ()
- stdio_sync_filebuf< char >::int_type **syncgetc** ()
- stdio_sync_filebuf< wchar_t >::int_type **syncgetc** ()
- int_type **syncputc** (int_type __c)
- stdio_sync_filebuf< char >::int_type **syncputc** (int_type __c)
- stdio_sync_filebuf< wchar_t >::int_type **syncputc** (int_type __c)
- int_type **syncungetc** (int_type __c)
- stdio_sync_filebuf< char >::int_type **syncungetc** (int_type __c)
- stdio_sync_filebuf< wchar_t >::int_type **syncungetc** (int_type __c)
- virtual int_type uflow ()
- virtual int_type underflow ()
- std::streamsize **xsgetn** (char *__s, std::streamsize __n)
- virtual std::streamsize **xsgetn** (char_type *__s, std::streamsize __n)
- std::streamsize **xsgetn** (wchar_t *__s, std::streamsize __n)
- virtual streamsize xsgetn (char_type *__s, streamsize __n)
- std::streamsize **xsputn** (const char *__s, std::streamsize __n)
- virtual std::streamsize **xsputn** (const char_type *__s, std::streamsize __n)
- std::streamsize **xsputn** (const wchar_t *__s, std::streamsize __n)
- virtual streamsize xsputn (const char_type *__s, streamsize __n)

- char_type * eback () const
- char_type * gptr () const

- char_type * egptr () const

- char_type * pbase () const
- char_type * pptr () const
- char_type * epptr () const

Protected Attributes

- locale _M_buf_locale
- char_type * _M_in_beg
- char_type * _M_in_cur
- char_type * _M_in_end
- char_type * _M_out_beg
- char_type * _M_out_cur
- char_type * _M_out_end

5.903.1 Detailed Description

template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
class __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >

Provides a layer of compatibility for C.
This GNU extension provides extensions for working with standard C FILE*'s. It must be instantiated by the user with the type of character used in the file stream, e.g., stdio_filebuf<char>.

5.903.2 Member Function Documentation

eback()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::eback () const  [inline], [protected], [inherited]
```

Access to the get area.
These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_stringbuf< _Cha std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_filebuf< _CharT, _Traits >::underflow(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

egptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::egptr () const  [inline], [protected], [inherited]
```

Access to the get area.
These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence

- egptr() returns the end pointer for the input sequence

Referenced by std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_filebuf< _CharT, _Traits >::underflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), std::wbuffer_convert< _Codecvt, _Elem, _Tr >::underflow(), std::basic_filebuf< _CharT, _Traits >::xsgetn(), and xsgetn().

epptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::epptr () const  [inline], [protected], [inherited]
```

Access to the put area.
These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_filebuf< _CharT, _Traits >::xsputn(), and xsputn().

file()

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
std::__c_file * __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::file ()  [inline]
```

Returns

The underlying FILE*.

This function can be used to access the underlying C file pointer. Note that there is no way for the library to track what you do with the file, so be careful.

gbump()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::gbump (
            int __n)  [inline], [protected], [inherited]
```

Moving the read position.

Parameters

_↩ _n	The delta by which to move.

This just advances the read position without returning any data.
Referenced by std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

getloc()

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::getloc () const  [inline], [inherited]
```

Locale access.

Returns

The current locale in effect.

If pubimbue(loc) has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

gptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::gptr () const  [inline], [protected], [inherited]
```

Access to the get area.
These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_stringbuf< _Cha std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_filebuf< _CharT, _Traits >::underflow(), std::basic_stringbuf< _CharT, _T std::wbuffer_convert< _Codecvt, _Elem, _Tr >::underflow(), std::basic_filebuf< _CharT, _Traits >::xsgetn(), and xsgetn().

imbue()

```
template<typename _CharT, typename _Traits>
virtual void std::basic_streambuf< _CharT, _Traits >::imbue (
            const locale & __loc)  [inline], [protected], [virtual], [inherited]
```

Changes translations.

Parameters

__loc	A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

Note

Base class version does nothing.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< _CharT, std::char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< wchar_t >, and std::basic_filebuf< wchar_t >.

in_avail()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::in_avail ()  [inline], [inherited]
```

Looking ahead into the stream.

Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

overflow()

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
virtual int_type __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::overflow (
            int_type __c = traits_type::eof())  [inline], [protected], [virtual]
```

Consumes data from the buffer; writes to the controlled sequence.

Parameters

_↩ _c	An additional character to consume.

Returns

eof() to indicate failure, something else (usually __c, or not_eof())

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if __c is not `eof()`.
For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.
A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

pbackfail()

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
virtual int_type __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::pbackfail (
            int_type __c = traits_type::eof())  [inline], [protected], [virtual]
```

Tries to back up the input sequence.

Parameters

_↩ _c	The character to be inserted back into the sequence.

Returns

eof() on failure, *some other value* on success

Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

pbase()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::pbase () const  [inline], [protected], [inherited]
```

Access to the put area.
These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, std::basic_filebuf< _CharT, _Traits >::sync(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

pbump()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::pbump (
            int __n)  [inline], [protected], [inherited]
```

Moving the write position.

Parameters

_↩ _n	The delta by which to move.

This just advances the write position without returning any data.
Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow().

pptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::pptr () const  [inline], [protected], [inherited]
```

Access to the put area.
These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_filebuf< _CharT, _T std::basic_filebuf< _CharT, _Traits >::xsputn(), and xsputn().

pubimbue()

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::pubimbue (
            const locale & __loc)  [inline], [inherited]
```

Entry point for imbue().

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls the derived imbue(__loc).

pubseekoff()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekoff (
            off_type __off,
            ios_base::seekdir __way,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [inline], [inherited]
```

Alters the stream position.

Parameters

__off	Offset.
__way	Value for ios_base::seekdir.
__mode	Value for ios_base::openmode.

Calls virtual seekoff function.

pubseekpos()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekpos (
            pos_type __sp,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [inline], [inherited]
```

Alters the stream position.

Parameters

__sp	Position
__mode	Value for ios_base::openmode.

Calls virtual seekpos function.

pubsetbuf()

```
template<typename _CharT, typename _Traits>
basic_streambuf * std::basic_streambuf< _CharT, _Traits >::pubsetbuf (
```

```
            char_type * __s,
            streamsize __n)  [inline], [inherited]
```

Entry points for derived buffer functions.
The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

pubsync()

```
template<typename _CharT, typename _Traits>
int std::basic_streambuf< _CharT, _Traits >::pubsync ()  [inline], [inherited]
```

Calls virtual sync function.
Referenced by std::basic_istream< _CharT, _Traits >::sync().

sbumpc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sbumpc ()  [inline], [inherited]
```

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.
Referenced by std::basic_istream< _CharT, _Traits >::getline().

seekoff()

```
template<typename _CharT, typename _Traits>
virtual pos_type std::basic_streambuf< _CharT, _Traits >::seekoff (
            off_type ,
            ios_base::seekdir ,
            ios_base::openmode  = ios_base::in | ios_base::out)  [inline], [protected], [virtual],
[inherited]
```

Alters the stream positions.
Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< char_type, traits_
std::basic_filebuf< wchar_t >, std::basic_filebuf< wchar_t >, std::basic_stringbuf< _CharT, _Traits, _Alloc >, std::basic_stringbuf< char >, std::basic_stringbuf< char >, std::basic_stringbuf< wchar_t >, and std::basic_stringbuf< wchar_t >.

seekpos()

```
template<typename _CharT, typename _Traits>
virtual pos_type std::basic_streambuf< _CharT, _Traits >::seekpos (
            pos_type ,
            ios_base::openmode  = ios_base::in | ios_base::out)  [inline], [protected], [virtual],
[inherited]
```

Alters the stream positions.
Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< char_type, traits_
std::basic_filebuf< wchar_t >, std::basic_filebuf< wchar_t >, std::basic_stringbuf< _CharT, _Traits, _Alloc >, std::basic_stringbuf< char >, std::basic_stringbuf< char >, std::basic_stringbuf< wchar_t >, and std::basic_stringbuf< wchar_t >.

setbuf()

```
template<typename _CharT, typename _Traits>
virtual basic_streambuf< char_type, _Traits > * std::basic_streambuf< _CharT, _Traits >::setbuf
(
            char_type * ,
            streamsize )  [inline], [protected], [virtual], [inherited]
```

Manipulates the buffer.
Each derived class provides its own appropriate behavior. See the next-to-last paragraph of https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html#io.streambuf.buffering for more on this function.

Note

Base class version does nothing, returns `this`.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< _CharT, std::char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< wchar_t >, std::basic_stringbuf< _CharT, _Traits, _Alloc >, std::basic_stringbuf< char >, and std::basic_stringbuf< char >.

setg()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setg (
            char_type * __gbeg,
            char_type * __gnext,
            char_type * __gend)  [inline], [protected], [inherited]
```

Setting the three read area pointers.

Parameters

__gbeg	A pointer.
__gnext	A pointer.
__gend	A pointer.

Postcondition

__gbeg == `eback()`, *__gnext* == `gptr()`, and *__gend* == `egptr()`

Referenced by std::wbuffer_convert< _Codecvt, _Elem, _Tr >::wbuffer_convert(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflo
std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

setp()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setp (
```

```
            char_type * __pbeg,
            char_type * __pend)  [inline], [protected], [inherited]
```

Setting the three write area pointers.

Parameters

__pbeg	A pointer.
__pend	A pointer.

Postcondition

__pbeg == `pbase()`, *__pbeg* == `pptr()`, and *__pend* == `epptr()`

Referenced by std::wbuffer_convert< _Codecvt, _Elem, _Tr >::wbuffer_convert().

sgetc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sgetc ()  [inline], [inherited]
```

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.
Referenced by std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), and std::basic_istream< _CharT, _Traits >::ignore().

sgetn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sgetn (
            char_type * __s,
            streamsize __n)  [inline], [inherited]
```

Entry point for xsgetn.

Parameters

__s	A buffer area.
__n	A count.

Returns xsgetn(__s,__n). The effect is to fill __*s*[0] through __*s*[__n-1] with characters from the input sequence, if possible.

showmanyc()

```
template<typename _CharT, typename _Traits>
virtual streamsize std::basic_streambuf< _CharT, _Traits >::showmanyc ()  [inline], [protected],
[virtual], [inherited]
```

Investigating the data available.

Returns

An estimate of the number of characters available in the input sequence, or -1.

If it returns a positive value, then successive calls to `underflow()` *will not return* `traits::eof()` *until at least that number of characters have been supplied. If* `showmanyc()` *returns -1, then calls to* `underflow()` *or* `uflow()` *will fail.* [27.5.2.4.3]/1

Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to underflow or uflow] will not return* `eof()` *but that they will return immediately.*

The standard adds that *the morphemes of* `showmanyc` *are* ***es-how-many-see****, not* ***show-manic****.*

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< _CharT, std::char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< wchar_t >, std::basic_filebuf< wchar_t >, std::basic_stringbuf< _CharT, _Traits, _Alloc >, std::basic_stringbuf< char >, std::basic_stringbuf< char std::basic_stringbuf< wchar_t >, and std::basic_stringbuf< wchar_t >.

snextc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::snextc ()  [inline], [inherited]
```

Getting the next character.

Returns

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.
Referenced by std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< char_type, traits_type >::ignore(), std::basic_istream< char_type, traits_type >::operator>>(), and std::basic_istream< char_type, traits_type >::putback().

sputbackc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputbackc (
            char_type __c)  [inline], [inherited]
```

Pushing characters back into the input stream.

Parameters

_↩ _c	The character to push back.

Returns

The previous character, if possible.

Similar to sungetc(), but __c is pushed onto the stream instead of *the previous character.* If successful, the next character fetched from the input stream will be __c.
Referenced by std::basic_istream< _CharT, _Traits >::putback().

sputc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputc (
             char_type __c)  [inline], [inherited]
```

Entry point for all single-character output functions.

Parameters

_↩ _c	A character to output.

Returns

__c, if possible.

One of two public output functions.
If a write position is available for the output sequence (i.e., the buffer is not full), stores __c in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(↩ __c)`.
Referenced by std::wbuffer_convert< _Codecvt, _Elem, _Tr >::overflow().

sputn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sputn (
            const char_type * __s,
            streamsize __n)  [inline], [inherited]
```

Entry point for all single-character output functions.

Parameters

_↩ _s	A buffer read area.
_↩ _n	A count.

One of two public output functions.
Returns xsputn(__s,__n). The effect is to write __s[0] through __s[__n-1] to the output sequence, if possible.

sungetc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sungetc ()  [inline], [inherited]
```

Moving backwards in the input stream.

Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns pbackfail(). The effect is to *unget* the last character *gotten*.
Referenced by std::basic_istream< char_type, traits_type >::sentry::sentry().

sync()

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
virtual int __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::sync ()  [inline], [protected],
[virtual]
```

Synchronizes the buffer arrays with the controlled sequences.

Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

Note

Base class version does nothing, returns zero.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

uflow()

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
virtual int_type __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::uflow ()  [inline], [protected],
[virtual]
```

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.
Reimplemented from std::basic_streambuf< _CharT, _Traits >.

underflow()

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
virtual int_type __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::underflow ()  [inline], [protected],
[virtual]
```

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.
For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.
A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

xsgetn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::xsgetn (
            char_type * __s,
            streamsize __n)  [protected], [virtual], [inherited]
```

Multiple character extraction.

Parameters

_↩ _s	A buffer area.
_↩ _n	Maximum number of characters to assign.

Returns

The number of characters assigned.

Fills __s[0] through __s[__n-1] with characters from the input sequence, as if by `sbumpc()`. Stops when either __n characters have been copied, or when `traits::eof()` would be copied.
It is expected that derived classes provide a more efficient implementation by overriding this definition.
Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< _CharT, std::char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< char_type, traits_type >, and std::basic_filebuf< wchar_t >.
References egptr(), gptr(), std::min(), and uflow().

xsputn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::xsputn (
            const char_type * __s,
            streamsize __n)  [protected], [virtual], [inherited]
```

Multiple character insertion.

Parameters

_↩ _s	A buffer area.
_↩ _n	Maximum number of characters to write.

Returns

The number of characters written.

Writes __s[0] through __s[__n-1] to the output sequence, as if by `sputc()`. Stops when either *n* characters have been copied, or when `sputc()` would return `traits::eof()`.
It is expected that derived classes provide a more efficient implementation by overriding this definition.
Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< _CharT, std::char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< char_type, traits_type >, and std::basic_filebuf< wchar_t >.
References epptr(), std::min(), overflow(), and pptr().

5.903.3 Member Data Documentation

_M_buf_locale

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::_M_buf_locale  [protected], [inherited]
```

Current locale setting.
Referenced by std::basic_filebuf< _CharT, _Traits >::basic_filebuf().

_M_in_beg

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_beg  [protected], [inherited]
```
Start of get area.

_M_in_cur

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_cur  [protected], [inherited]
```
Current read area.

_M_in_end

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_end  [protected], [inherited]
```
End of get area.

_M_out_beg

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_beg  [protected], [inherited]
```
Start of put area.

_M_out_cur

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_cur  [protected], [inherited]
```
Current put area.

_M_out_end

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_end  [protected], [inherited]
```
End of put area.
The documentation for this class was generated from the following file:

- stdio_sync_filebuf.h

5.904 std::chrono::steady_clock Struct Reference

```
#include <chrono.h>
```

Public Types

- typedef chrono::nanoseconds **duration**
- typedef duration::period **period**
- typedef duration::rep **rep**
- typedef chrono::time_point< steady_clock, duration > **time_point**

Static Public Member Functions

- static time_point **now** () noexcept

Static Public Attributes

- static constexpr bool **is_steady**

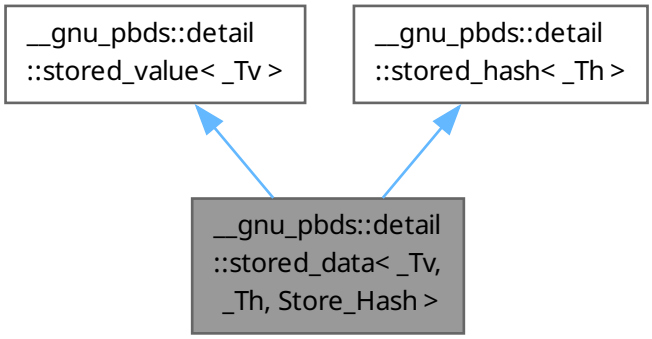
5.904.1 Detailed Description

Monotonic clock.
Time returned has the property of only increasing at a uniform rate.
The documentation for this struct was generated from the following file:

- chrono.h

5.905 __gnu_pbds::detail::stored_data< _Tv, _Th, Store_Hash > Struct Template Reference

`#include <types_traits.hpp>`
Inheritance diagram for __gnu_pbds::detail::stored_data< _Tv, _Th, Store_Hash >:



Public Types

- typedef _Th **hash_type**
- typedef _Tv **value_type**

Public Attributes

- hash_type **m_hash**
- value_type **m_value**

5.905.1 Detailed Description

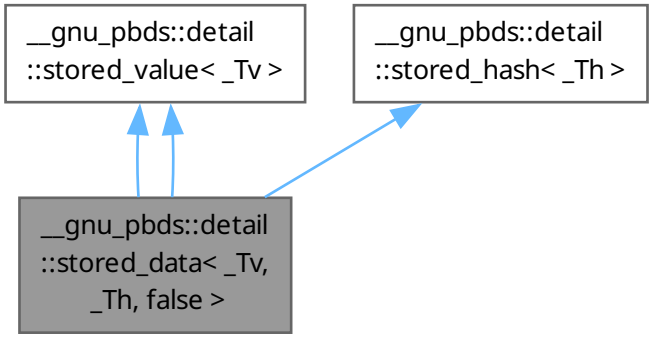
template<typename _Tv, typename _Th, bool Store_Hash>
struct __gnu_pbds::detail::stored_data< _Tv, _Th, Store_Hash >

Primary template for representation of stored data. Two types of data can be stored: value and hash.
The documentation for this struct was generated from the following file:

- types_traits.hpp

5.906 __gnu_pbds::detail::stored_data< _Tv, _Th, false > Struct Template Reference

`#include <types_traits.hpp>`
Inheritance diagram for __gnu_pbds::detail::stored_data< _Tv, _Th, false >:



Public Types

- typedef _Th **hash_type**
- typedef _Tv **value_type**

Public Attributes

- hash_type **m_hash**
- value_type **m_value**

5.906.1 Detailed Description

template<**typename _Tv, typename _Th**>
struct __gnu_pbds::detail::stored_data< **_Tv, _Th, false** >

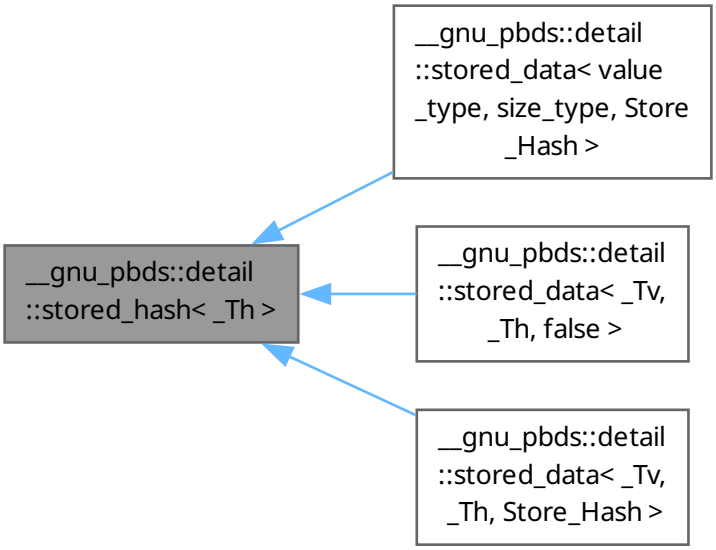
Specialization for representation of stored data of just value type.
The documentation for this struct was generated from the following file:

- types_traits.hpp

5.907 __gnu_pbds::detail::stored_hash< _Th > Struct Template Reference

`#include <types_traits.hpp>`

Inheritance diagram for __gnu_pbds::detail::stored_hash< _Th >:



Public Types

- typedef _Th **hash_type**

Public Attributes

- hash_type **m_hash**

5.907.1 Detailed Description

template<typename _Th>
struct __gnu_pbds::detail::stored_hash< _Th >

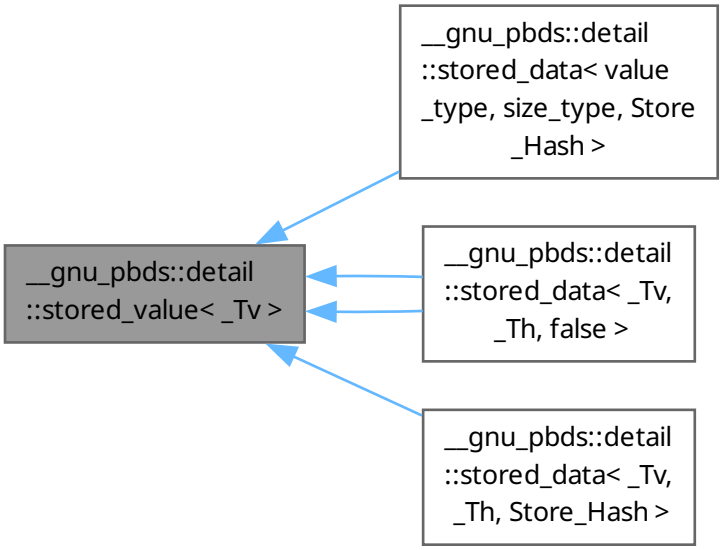
Stored hash.
The documentation for this struct was generated from the following file:

- types_traits.hpp

5.908 __gnu_pbds::detail::stored_value< _Tv > Struct Template Reference

```
#include <types_traits.hpp>
```

Inheritance diagram for __gnu_pbds::detail::stored_value< _Tv >:



Public Types

- typedef _Tv **value_type**

Public Attributes

- value_type **m_value**

5.908.1 Detailed Description

template<**typename _Tv**>
struct __gnu_pbds::detail::stored_value< **_Tv** >

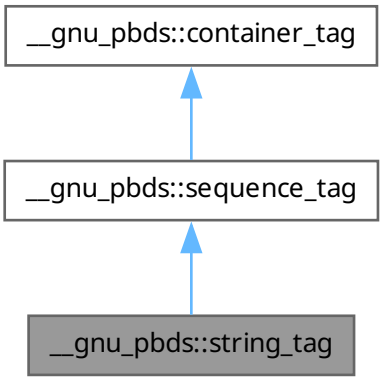
Stored value.
The documentation for this struct was generated from the following file:

- types_traits.hpp

5.909 __gnu_pbds::string_tag Struct Reference

`#include <tag_and_trait.hpp>`

Inheritance diagram for __gnu_pbds::string_tag:



5.909.1 Detailed Description

Basic string container, inclusive of strings, ropes, etc.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

5.910 std::student_t_distribution< _RealType > Class Template Reference

`#include <random>`

Classes

- struct param_type

Public Types

- typedef _RealType result_type

Public Member Functions

- **student_t_distribution** (_RealType __n)
- **student_t_distribution** (const param_type &__p)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)

- result_type max () const
- result_type min () const
- _RealType **n** () const
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- template<typename _RealType1, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::student_t_distribution< _RealType1 > &__x)
- bool operator== (const student_t_distribution &__d1, const student_t_distribution &__d2)
- template<typename _RealType1, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::student_t_distribution< _RealType1 > &__x)

5.910.1 Detailed Description

template<typename _RealType = double>
class std::student_t_distribution< _RealType >

A student_t_distribution random number distribution.
The formula for the normal probability mass function is:

$$p(x|n) = \frac{1}{\sqrt{(n\pi)}} \frac{\Gamma((n+1)/2)}{\Gamma(n/2)} (1 + \frac{x^2}{n})^{-(n+1)/2}$$

Since

C++11

5.910.2 Member Typedef Documentation

result_type

```
template<typename _RealType = double>
typedef _RealType std::student_t_distribution< _RealType >::result_type
```

The type of the range of the distribution.

5.910.3 Member Function Documentation

max()

```
template<typename _RealType = double>
result_type std::student_t_distribution< _RealType >::max () const  [inline]
```

Returns the least upper bound value of the distribution.

min()

```
template<typename _RealType = double>
result_type std::student_t_distribution< _RealType >::min () const  [inline]
```

Returns the greatest lower bound value of the distribution.

operator()()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::student_t_distribution< _RealType >::operator() (
            _UniformRandomNumberGenerator & __urng)  [inline]
```

Generating functions.

param() [1/2]

```
template<typename _RealType = double>
param_type std::student_t_distribution< _RealType >::param () const  [inline]
```

Returns the parameter set of the distribution.

param() [2/2]

```
template<typename _RealType = double>
void std::student_t_distribution< _RealType >::param (
            const param_type & __param)  [inline]
```

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

reset()

```
template<typename _RealType = double>
void std::student_t_distribution< _RealType >::reset ()  [inline]
```

Resets the distribution state.

5.910.4 Friends And Related Symbol Documentation

operator<<

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::student_t_distribution< _RealType1 > & __x)  [friend]
```

Inserts a student_t_distribution random number distribution __x into the output stream __os.

Parameters

__os	An output stream.
__x	A student_t_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

operator==

```
template<typename _RealType = double>
bool operator== (
             const student_t_distribution< _RealType > & __d1,
             const student_t_distribution< _RealType > & __d2)  [friend]
```

Return true if two Student t distributions have the same parameters and the sequences that would be generated are equal.

operator>>

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
             std::basic_istream< _CharT, _Traits > & __is,
             std::student_t_distribution< _RealType1 > & __x)  [friend]
```

Extracts a student_t_distribution random number distribution __x from the input stream __is.

Parameters

_↩ _is	An input stream.
_↩ _x	A student_t_distribution random number generator engine.

Returns

The input stream with __x extracted or in an error state.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

5.911 std::sub_match< _BiIter > Class Template Reference

```
#include <regex>
```

Public Types

- typedef __iter_traits::difference_type **difference_type**
- typedef _BiIter **iterator**
- typedef basic_string< value_type > **string_type**
- typedef __iter_traits::value_type **value_type**

Public Member Functions

- int compare (const sub_match &__s) const
- difference_type length () const noexcept
- operator string_type () const
- string_type str () const
- void swap (sub_match &__s) noexcept(__is_nothrow_swappable< _BiIter >::value)

- int compare (const string_type &__s) const
- int compare (const value_type *__s) const

Public Attributes

- iterator **first**
- bool **matched**
- iterator **second**

Related Symbols

(Note that these are not member symbols.)

- template<typename _BiIter>
  bool operator== (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__rhs)
- template<typename _BiIter>
  auto operator<=> (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__rhs) noexcept(__↩
  detail::__is_contiguous_iter< _BiIter >::value)
- template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc>
  bool operator== (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch↩
  _alloc > &__rhs)
- template<typename _Bi_iter, typename _Ch_traits, typename _Alloc>
  auto operator<=> (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _↩
  Alloc > &__rhs) noexcept(__detail::__is_contiguous_iter< _Bi_iter >::value)
- template<typename _Bi_iter>
  bool operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const
  *__rhs)
- template<typename _Bi_iter>
  auto operator<=> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const
  *__rhs) noexcept(__detail::__is_contiguous_iter< _Bi_iter >::value)
- template<typename _Bi_iter>
  bool operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const
  &__rhs)
- template<typename _Bi_iter>
  auto operator<=> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const
  &__rhs) noexcept(__detail::__is_contiguous_iter< _Bi_iter >::value)
- template<typename _Ch_type, typename _Ch_traits, typename _Bi_iter>
  basic_ostream< _Ch_type, _Ch_traits > & operator<< (basic_ostream< _Ch_type, _Ch_traits > &__os, const
  sub_match< _Bi_iter > &__m)

5.911.1 Detailed Description

template<typename _BiIter>
class std::sub_match< _BiIter >

A sequence of characters matched by a particular marked sub-expression.
An object of this class is essentially a pair of iterators marking a matched subexpression within a regular expression pattern match. Such objects can be converted to and compared with std::basic_string objects of the same character type as the pattern matched by the regular expression.
A `sub_match<Iter>` has a public base class of type `pair<Iter, Iter>`, so inherits pair's data members named `first` and `second`. The iterators that make up the pair are the usual half-open interval referencing the actual original pattern matched.

Since

C++11

5.911.2 Member Function Documentation

compare() [1/3]

```
template<typename _BiIter>
int std::sub_match< _BiIter >::compare (
            const string_type & __s) const  [inline]
```

Compares this sub_match to a string.

Parameters

_↩ _s	A string to compare to this sub_match.

Return values

negative	This matched sequence will collate before __s.
zero	This matched sequence is equivalent to __s.
positive	This matched sequence will collate after __s.

compare() [2/3]

```
template<typename _BiIter>
int std::sub_match< _BiIter >::compare (
            const sub_match< _BiIter > & __s) const  [inline]
```

Compares this and another matched sequence.

Parameters

_↩ _s	Another matched sequence to compare to this one.

Return values

negative	This matched sequence will collate before __s.
zero	This matched sequence is equivalent to __s.
positive	This matched sequence will collate after __s.

Referenced by std::sub_match< const char * >::operator==(), and std::sub_match< const char * >::operator==().

compare() [3/3]

```
template<typename _BiIter>
int std::sub_match< _BiIter >::compare (
            const value_type * __s) const  [inline]
```

Compares this sub_match to a string.

Parameters

_↩ _s	A string to compare to this sub_match.

Return values

negative	This matched sequence will collate before __s.
zero	This matched sequence is equivalent to __s.
positive	This matched sequence will collate after __s.

length()

```
template<typename _BiIter>
difference_type std::sub_match< _BiIter >::length () const  [inline], [noexcept]
```

Gets the length of the matching sequence.

operator string_type()

```
template<typename _BiIter>
std::sub_match< _BiIter >::operator string_type () const  [inline]
```

Gets the matching sequence as a string.

Returns

the matching sequence as a string.

This is the implicit conversion operator. It is identical to the str() member function except that it will want to pop up in unexpected places and cause a great deal of confusion and cursing from the unwary.

str()

```
template<typename _BiIter>
string_type std::sub_match< _BiIter >::str () const  [inline]
```

Gets the matching sequence as a string.

Returns

the matching sequence as a string.

Referenced by std::sub_match< const char * >::operator string_type().

swap()

```
template<typename _BiIter>
void std::sub_match< _BiIter >::swap (
            sub_match< _BiIter > & __s)  [inline], [noexcept]
```

Swap the values of two sub_match objects.

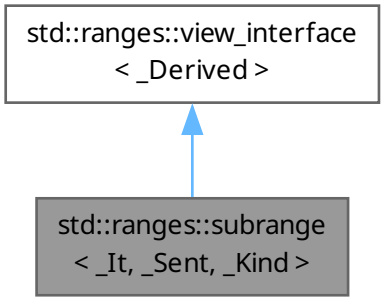
The documentation for this class was generated from the following file:

- regex.h

5.912 std::ranges::subrange< _It, _Sent, _Kind > Class Template Reference

```
#include <ranges_util.h>
```

Inheritance diagram for std::ranges::subrange< _It, _Sent, _Kind >:



Public Member Functions

- constexpr **subrange** (__detail::__convertible_to_non_slicing< _It > auto __i, _Sent __s) noexcept(is_nothrow↩ _constructible_v< _It, decltype(__i)> &&is_nothrow_constructible_v< _Sent, _Sent & >)
- constexpr **subrange** (__detail::__convertible_to_non_slicing< _It > auto __i, _Sent __s, __size_type __↩ n) noexcept(is_nothrow_constructible_v< _It, decltype(__i)> &&is_nothrow_constructible_v< _Sent, _Sent & >)
- template<__detail::__different_from< subrange > _Rng>
  requires borrowed_range<_Rng> && __detail::__convertible_to_non_slicing<iterator_t<_Rng>, _It> && convertible_to<sentinel_t<_↩ Rng>, _Sent>
  constexpr **subrange** (_Rng &&__r) noexcept(noexcept(subrange(__r, ranges::size(__r)))) _S_store_size &&sized_range< _Rng >
- template<__detail::__different_from< subrange > _Rng>
  requires borrowed_range<_Rng> && __detail::__convertible_to_non_slicing<iterator_t<_Rng>, _It> && convertible_to<sentinel_t<_↩ Rng>, _Sent> (!_S_store_size)
  constexpr **subrange** (_Rng &&__r) noexcept(noexcept(subrange(ranges::begin(__r), ranges::end(__r))))
- template<borrowed_range _Rng>
  requires __detail::__convertible_to_non_slicing<iterator_t<_Rng>, _It> && convertible_to<sentinel_t<_Rng>, _Sent> (_Kind == subrange_kind::sized)
  constexpr **subrange** (_Rng &&__r, __size_type __n) noexcept(noexcept(subrange(ranges::begin(__r), ranges↩ ::end(__r), __n)))
- constexpr subrange & **advance** (iter_difference_t< _It > __n)
- constexpr decltype(auto) **back** ()
- constexpr decltype(auto) **back** () const
- constexpr _It **begin** ()
- constexpr _It **begin** () const
- constexpr auto **data** () const noexcept(noexcept(ranges::begin(_M_derived())))
- constexpr auto **data** () noexcept(noexcept(ranges::begin(_M_derived())))
- constexpr bool **empty** () const
- constexpr bool **empty** () noexcept(noexcept(_S_empty(_M_derived())))
- constexpr _Sent **end** () const
- constexpr decltype(auto) **front** ()
- constexpr decltype(auto) **front** () const

- constexpr subrange **next** (iter_difference_t< _It > __n=1) &&
- constexpr subrange **next** (iter_difference_t< _It > __n=1) const &
- template<__detail::__different_from< subrange > _PairLike>
  requires __detail::__pair_like_convertible_from<_PairLike, const _It&, const _Sent&>
  constexpr **operator _PairLike** () const
- constexpr **operator bool** () const noexcept(noexcept(ranges::empty(_M_derived())))
- constexpr **operator bool** () noexcept(noexcept(ranges::empty(_M_derived())))
- template<random_access_range _Range = _Derived>
  constexpr decltype(auto) **operator[ ]** (range_difference_t< _Range > __n)
- template<random_access_range _Range = const _Derived>
  constexpr decltype(auto) **operator[ ]** (range_difference_t< _Range > __n) const
- constexpr subrange **prev** (iter_difference_t< _It > __n=1) const
- constexpr __size_type **size** () const
- constexpr auto **size** () noexcept(noexcept(_S_size(_M_derived())))

Friends

- struct **views::_Drop**

5.912.1 Detailed Description

template<input_or_output_iterator _It, sentinel_for< _It > _Sent = _It, subrange_kind _Kind = sized_sentinel↩ _for<_Sent, _It> ? subrange_kind::sized : subrange_kind::unsized>
requires (_Kind == subrange_kind::sized || !sized_sentinel_for<_Sent, _It>)
class std::ranges::subrange< _It, _Sent, _Kind >

The ranges::subrange class template.
The documentation for this class was generated from the following file:

- ranges_util.h

5.913 std::subtract_with_carry_engine< _UIntType, __w, __s, __r > Class Template Reference

`#include <random>`

Public Types

- typedef _UIntType result_type

Public Member Functions

- template<typename _Sseq, typename = _If_seed_seq<_Sseq>>
  subtract_with_carry_engine (_Sseq &__q)
- subtract_with_carry_engine (result_type __sd)
- void discard (unsigned long long __z)
- result_type operator() ()
- template<typename _Sseq>
  _If_seed_seq< _Sseq > seed (_Sseq &__q)
- template<typename _Sseq>
  auto **seed** (_Sseq &__q) -> _If_seed_seq< _Sseq >
- void seed (result_type __sd=0u)

Static Public Member Functions

- static constexpr result_type max ()
- static constexpr result_type min ()

Static Public Attributes

- static constexpr uint_least32_t **default_seed**
- static constexpr size_t **long_lag**
- static constexpr size_t **short_lag**
- static constexpr size_t **word_size**

Friends

- template<typename _UIntType1, size_t __w1, size_t __s1, size_t __r1, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::subtract_with_carry_engine< _UIntType1, __w1, __s1, __r1 > &__x)
- bool operator== (const subtract_with_carry_engine &__lhs, const subtract_with_carry_engine &__rhs)
- template<typename _UIntType1, size_t __w1, size_t __s1, size_t __r1, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::subtract_with_carry_engine< _UIntType1, __w1, __s1, __r1 > &__x)

5.913.1 Detailed Description

template<typename _UIntType, size_t __w, size_t __s, size_t __r>
class std::subtract_with_carry_engine< _UIntType, __w, __s, __r >

The Marsaglia-Zaman generator.
This is a model of a Generalized Fibonacci discrete random number generator, sometimes referred to as the SWC generator.
A discrete random number generator that produces pseudorandom numbers using:

$$x_i \leftarrow (x_{i-s} - x_{i-r} - carry_{i-1}) \bmod m$$

The size of the state is r and the maximum period of the generator is $(m^r - m^s - 1)$.

Since

C++11

5.913.2 Member Typedef Documentation

result_type

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
typedef _UIntType std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::result_type
```

The type of the generated random value.

5.913.3 Constructor & Destructor Documentation

subtract_with_carry_engine() [1/2]

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::subtract_with_carry_engine (
            result_type __sd)  [inline], [explicit]
```

Constructs an explicitly seeded subtract_with_carry_engine random number generator.

subtract_with_carry_engine() [2/2]

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
template<typename _Sseq, typename = _If_seed_seq<_Sseq>>
std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::subtract_with_carry_engine (
            _Sseq & __q)  [inline], [explicit]
```

Constructs a subtract_with_carry_engine random number engine seeded from the seed sequence __q.

Parameters

_↩ _q	the seed sequence.

5.913.4 Member Function Documentation

discard()

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
void std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::discard (
            unsigned long long __z)  [inline]
```

Discard a sequence of random numbers.

max()

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
static constexpr result_type std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::max ()
[inline], [static], [constexpr]
```

Gets the inclusive maximum value of the range of random integers returned by this generator.

min()

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
static constexpr result_type std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::min ()
[inline], [static], [constexpr]
```

Gets the inclusive minimum value of the range of random integers returned by this generator.

operator()()

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
subtract_with_carry_engine< _UIntType, __w, __s, __r >::result_type std::subtract_with_carry_engine<
_UIntType, __w, __s, __r >::operator() ()
```

Gets the next random number in the sequence.

seed() [1/2]

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
template<typename _Sseq>
_If_seed_seq< _Sseq > std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::seed (
            _Sseq & __q)
```

Seeds the initial state x_0 of the % subtract_with_carry_engine random number generator.

seed() [2/2]

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
void std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::seed (
            result_type __sd = 0u)
```

Seeds the initial state x_0 of the random number generator.
N1688[4.19] modifies this as follows. If __value == 0, sets value to 19780503. In any case, with a linear congruential generator lcg(i) having parameters $m_{lcg} = 2147483563, a_{lcg} = 40014, c_{lcg} = 0, and lcg(0) = value$, sets $x_{-r} \dots x_{-1}$ to $lcg(1) \bmod m \dots lcg(r) \bmod m$ respectively. If $x_{-1} = 0$ set carry to 1, otherwise sets carry to 0.
Referenced by std::subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 >::subtract_with_carry_engine(), and std::subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 >::subtract_with_carry_engine().

5.913.5 Friends And Related Symbol Documentation

operator<<

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
template<typename _UIntType1, size_t __w1, size_t __s1, size_t __r1, typename _CharT, typename _↩
Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::subtract_with_carry_engine< _UIntType1, __w1, __s1, __r1 > & __x)  [friend]
```

Inserts the current state of a % subtract_with_carry_engine random number generator engine __x into the output stream __os.

Parameters

__os	An output stream.
__x	A % subtract_with_carry_engine random number generator engine.

Returns

The output stream with the state of __x inserted or in an error state.

operator==

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
bool operator== (
            const subtract_with_carry_engine< _UIntType, __w, __s, __r > & __lhs,
            const subtract_with_carry_engine< _UIntType, __w, __s, __r > & __rhs)  [friend]
```

Compares two % subtract_with_carry_engine random number generator objects of the same type for equality.

Parameters

__lhs	A % subtract_with_carry_engine random number generator object.
__rhs	Another % subtract_with_carry_engine random number generator object.

Returns

true if the infinite sequences of generated values would be equal, false otherwise.

operator>>

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
template<typename _UIntType1, size_t __w1, size_t __s1, size_t __r1, typename _CharT, typename _↩
Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            std::subtract_with_carry_engine< _UIntType1, __w1, __s1, __r1 > & __x)  [friend]
```

Extracts the current state of a % subtract_with_carry_engine random number generator engine __x from the input stream __is.

Parameters

_↩ _is	An input stream.
_↩ _x	A % subtract_with_carry_engine random number generator engine.

Returns

The input stream with the state of __x extracted or in an error state.

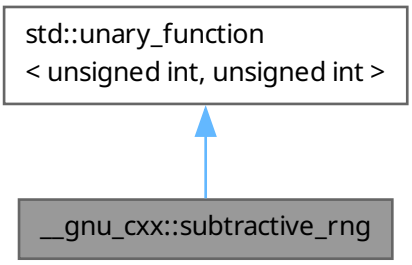
The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

5.914 __gnu_cxx::subtractive_rng Class Reference

`#include <functional>`

Inheritance diagram for __gnu_cxx::subtractive_rng:



Public Types

- typedef unsigned int argument_type
- typedef unsigned int result_type

Public Member Functions

- subtractive_rng ()
- subtractive_rng (unsigned int __seed)
- void **_M_initialize** (unsigned int __seed)
- unsigned int operator() (unsigned int __limit)

5.914.1 Detailed Description

The `subtractive_rng` class is documented on SGI's site. Note that this code assumes that `int` is 32 bits.

5.914.2 Member Typedef Documentation

argument_type

```
typedef unsigned int std::unary_function< unsigned int, unsigned int >::argument_type  [inherited]
```

`argument_type` is the type of the argument

result_type

```
typedef unsigned int std::unary_function< unsigned int, unsigned int >::result_type  [inherited]
```

`result_type` is the return type

5.914.3 Constructor & Destructor Documentation

subtractive_rng() [1/2]

```
__gnu_cxx::subtractive_rng::subtractive_rng (
            unsigned int __seed)  [inline]
```

Ctor allowing you to initialize the seed.

subtractive_rng() [2/2]

```
__gnu_cxx::subtractive_rng::subtractive_rng ()  [inline]
```

Default ctor; initializes its state with some number you don't see.

5.914.4 Member Function Documentation

operator()()

```
unsigned int __gnu_cxx::subtractive_rng::operator() (
            unsigned int __limit)  [inline]
```

Returns a number less than the argument.
The documentation for this class was generated from the following file:

- ext/functional

5.915 __gnu_pbds::detail::synth_access_traits< Type_Traits, Set, _ATraits > Struct Template Reference

`#include <synth_access_traits.hpp>`

Public Types

- typedef _ATraits **base_type**
- typedef base_type::const_iterator **const_iterator**
- typedef type_traits::const_reference **const_reference**

- typedef type_traits::key_const_reference **key_const_reference**
- typedef Type_Traits **type_traits**

Public Member Functions

- **synth_access_traits** (const base_type &)
- bool **cmp_keys** (key_const_reference, key_const_reference) const
- bool **cmp_prefixes** (const_iterator, const_iterator, const_iterator, const_iterator, bool compare_after=false) const
- bool **equal_keys** (key_const_reference, key_const_reference) const
- bool **equal_prefixes** (const_iterator, const_iterator, const_iterator, const_iterator, bool compare_after=true) const

Static Public Member Functions

- static key_const_reference **extract_key** (const_reference)

5.915.1 Detailed Description

template<typename Type_Traits, bool Set, typename _ATraits>
struct __gnu_pbds::detail::synth_access_traits< Type_Traits, Set, _ATraits >

Synthetic element access traits.
The documentation for this struct was generated from the following file:

- synth_access_traits.hpp

5.916 std::chrono::system_clock Struct Reference

`#include <chrono.h>`

Public Types

- typedef chrono::nanoseconds **duration**
- typedef duration::period **period**
- typedef duration::rep **rep**
- typedef chrono::time_point< system_clock, duration > **time_point**

Static Public Member Functions

- static time_point **from_time_t** (std::time_t __t) noexcept
- static time_point **now** () noexcept
- static std::time_t **to_time_t** (const time_point &__t) noexcept

Static Public Attributes

- static constexpr bool **is_steady**

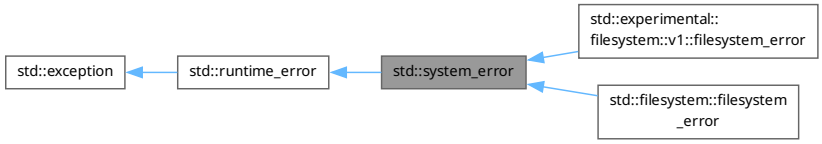
5.916.1 Detailed Description

System clock.
Time returned represents wall time from the system-wide clock.
The documentation for this struct was generated from the following file:

- chrono.h

5.917 std::system_error Class Reference

`#include <system_error>`
Inheritance diagram for std::system_error:



Public Member Functions

- **system_error** (const system_error &)=default
- **system_error** (error_code __ec, const char *__what)
- **system_error** (error_code __ec, const string &__what)
- **system_error** (error_code __ec=error_code())
- **system_error** (int __v, const error_category &__ecat)
- **system_error** (int __v, const error_category &__ecat, const char *__what)
- **system_error** (int __v, const error_category &__ecat, const string &__what)
- const error_code & **code** () const noexcept
- system_error & **operator=** (const system_error &)=default
- virtual const char * what () const noexcept

5.917.1 Detailed Description

An exception type that includes an `error_code` value.
Typically used to report errors from the operating system and other low-level APIs.

Since

C++11

5.917.2 Member Function Documentation

what()

`virtual const char * std::runtime_error::what () const [virtual], [noexcept], [inherited]`
Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).
Reimplemented from std::exception.
Reimplemented in std::experimental::filesystem::v1::filesystem_error, and std::filesystem::filesystem_error.
The documentation for this class was generated from the following file:

- system_error

5.918 std::chrono::tai_clock Class Reference

`#include <chrono>`

Public Types

- using **duration**
- using **period**
- using **rep**
- using **time_point**

Static Public Member Functions

- template<typename _Duration>
  static tai_time< common_type_t< _Duration, seconds > > **from_utc** (const utc_time< _Duration > &__t)
- static time_point **now** ()
- template<typename _Duration>
  static utc_time< common_type_t< _Duration, seconds > > **to_utc** (const tai_time< _Duration > &__t)

Static Public Attributes

- static constexpr bool **is_steady**

5.918.1 Detailed Description

A clock that measures International Atomic Time.
The epoch is 1958-01-01 00:00:00.

Since

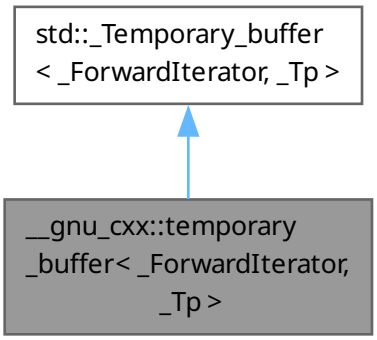
C++20

The documentation for this class was generated from the following file:

- chrono

5.919 __gnu_cxx::temporary_buffer< _ForwardIterator, _Tp > Struct Template Reference

`#include <memory>`
Inheritance diagram for __gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >:



Public Types

- typedef pointer **iterator**
- typedef value_type * **pointer**
- typedef ptrdiff_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- temporary_buffer (_ForwardIterator __first, _ForwardIterator __last)
- ~temporary_buffer ()
- size_type _M_requested_size () const
- iterator begin ()
- iterator end ()
- size_type size () const

Protected Attributes

- struct std::_Temporary_buffer::_Impl **_M_impl**
- size_type **_M_original_len**

5.919.1 Detailed Description

template<class _ForwardIterator, class _Tp = typename std::iterator_traits<_ForwardIterator>::value_type>
struct __gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >

This class provides similar behavior and semantics of the standard functions get_temporary_buffer() and return_↩ temporary_buffer(), but encapsulated in a type vaguely resembling a standard container.
By default, a temporary_buffer<Iter> stores space for objects of whatever type the Iter iterator points to. It is constructed from a typical [first,last) range, and provides the begin(), end(), size() functions, as well as requested_size(). For non-trivial types, copies of *first will be used to initialize the storage.
`malloc` is used to obtain underlying storage.
Like get_temporary_buffer(), not all the requested memory may be available. Ideally, the created buffer will be large enough to hold a copy of [first,last), but if size() is less than requested_size(), then this didn't happen.

5.919.2 Constructor & Destructor Documentation

temporary_buffer()

```
template<class _ForwardIterator, class _Tp = typename std::iterator_traits<_ForwardIterator>↩
::value_type>
__gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >::temporary_buffer (
            _ForwardIterator __first,
            _ForwardIterator __last)  [inline]
```

Requests storage large enough to hold a copy of [first,last).

~temporary_buffer()

```
template<class _ForwardIterator, class _Tp = typename std::iterator_traits<_ForwardIterator>↩
::value_type>
__gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >::~temporary_buffer ()  [inline]
```

Destroys objects and frees storage.

5.919.3 Member Function Documentation

_M_requested_size()

```
template<typename _ForwardIterator, typename _Tp>
size_type std::_Temporary_buffer< _ForwardIterator, _Tp >::_M_requested_size () const  [inline],
[inherited]
```

Returns the size requested by the constructor; may be >size().

begin()

```
template<typename _ForwardIterator, typename _Tp>
iterator std::_Temporary_buffer< _ForwardIterator, _Tp >::begin ()  [inline], [inherited]
```

As per Table mumble.
Referenced by _Temporary_buffer().

end()

```
template<typename _ForwardIterator, typename _Tp>
iterator std::_Temporary_buffer< _ForwardIterator, _Tp >::end ()  [inline], [inherited]
```

As per Table mumble.
Referenced by _Temporary_buffer().

size()

```
template<typename _ForwardIterator, typename _Tp>
size_type std::_Temporary_buffer< _ForwardIterator, _Tp >::size () const  [inline], [inherited]
```

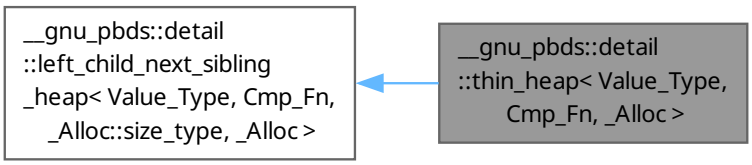
As per Table mumble.
The documentation for this struct was generated from the following file:

- ext/memory

5.920 __gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc > Class Template Reference

```
#include <thin_heap_.hpp>
```

Inheritance diagram for __gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc >:



Public Types

- typedef _Alloc **allocator_type**
- typedef Cmp_Fn **cmp_fn**
- typedef base_type::const_iterator **const_iterator**

- typedef __rebind_a::const_pointer **const_pointer**
- typedef __rebind_a::const_reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef base_type::iterator **iterator**
- typedef base_type::point_const_iterator **point_const_iterator**
- typedef base_type::point_iterator **point_iterator**
- typedef __rebind_a::pointer **pointer**
- typedef __rebind_a::reference **reference**
- typedef _Alloc::size_type **size_type**
- typedef Value_Type **value_type**

Public Member Functions

- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- void **erase** (point_iterator)
- template<typename Pred>
  size_type **erase_if** (Pred)
- Cmp_Fn & **get_cmp_fn** ()
- const Cmp_Fn & **get_cmp_fn** () const
- void **join** (thin_heap< Value_Type, Cmp_Fn, _Alloc > &)
- size_type **max_size** () const
- void **modify** (point_iterator, const_reference)
- void **pop** ()
- point_iterator **push** (const_reference)
- size_type **size** () const
- template<typename Pred>
  void **split** (Pred, thin_heap< Value_Type, Cmp_Fn, _Alloc > &)
- void **swap** (left_child_next_sibling_heap< Value_Type, Cmp_Fn, _Alloc::size_type, _Alloc > &)
- const_reference **top** () const

Protected Types

- typedef base_type::node **node**
- typedef alloc_traits::allocator_type **node_allocator**
- typedef base_type::node_const_pointer **node_const_pointer**
- typedef _Alloc::size_type **node_metadata**
- typedef base_type::node_pointer **node_pointer**
- typedef std::pair< node_pointer, node_pointer > **node_pointer_pair**

Protected Member Functions

- **thin_heap** (const Cmp_Fn &)
- **thin_heap** (const thin_heap< Value_Type, Cmp_Fn, _Alloc > &)
- void **actual_erase_node** (node_pointer)
- void **bubble_to_top** (node_pointer)
- void **clear_imp** (node_pointer)

- template<typename It>
  void **copy_from_range** (It, It)
- node_pointer **get_new_node_for_insert** (const_reference)
- node_pointer **prune** (Pred)
- void **swap** (thin_heap< Value_Type, Cmp_Fn, _Alloc > &)
- void **swap_with_parent** (node_pointer, node_pointer)
- void **to_linked_list** ()
- void **value_swap** (left_child_next_sibling_heap &)

Static Protected Member Functions

- static node_pointer **parent** (node_pointer)

Protected Attributes

- node_pointer **m_p_root**
- size_type **m_size**

5.920.1 Detailed Description

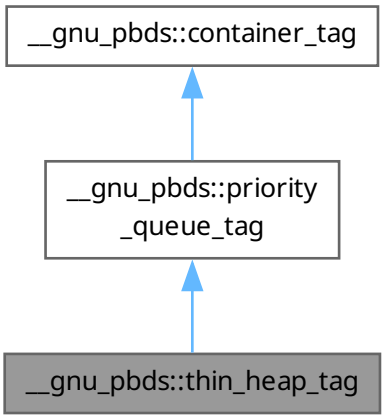
template<typename Value_Type, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc >

Thin heap.
See Tarjan and Kaplan.
The documentation for this class was generated from the following file:

- thin_heap_.hpp

5.921 __gnu_pbds::thin_heap_tag Struct Reference

`#include <tag_and_trait.hpp>`
Inheritance diagram for __gnu_pbds::thin_heap_tag:



5.921.1 Detailed Description

Thin heap.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

5.922 std::thread Class Reference

`#include <thread>`

Classes

- class id

Public Types

- using **native_handle_type**

Public Member Functions

- template<typename _Callable, typename... _Args, typename = _Require<__not_same<_Callable>>>
  thread (_Callable &&__f, _Args &&... __args)
- **thread** (const thread &)=delete
- **thread** (thread &&__t) noexcept
- void **detach** ()
- id **get_id** () const noexcept
- void **join** ()
- bool **joinable** () const noexcept
- native_handle_type native_handle ()
- thread & **operator=** (const thread &)=delete
- thread & **operator=** (thread &&__t) noexcept
- void **swap** (thread &__t) noexcept

Static Public Member Functions

- static unsigned int **hardware_concurrency** () noexcept

Related Symbols

(Note that these are not member symbols.)

- bool **operator==** (thread::id __x, thread::id __y) noexcept
- void **swap** (thread &__x, thread &__y) noexcept

5.922.1 Detailed Description

A std::thread represents a new thread of execution.
The default constructor creates an object that does not own a thread. The `thread(F&&, Args&&...)` constructor invokes a callable in a new thread, and owns that new thread. A `std::thread` that owns a thread is *joinable*. Joining a thread waits for it to finish executing, which happens when the callable running in that thread returns.
A `std::thread` cannot be copied, but can be moved. Moving a joinable object transfers ownership of its thread to another object.
A joinable `std::thread` must be explicitly joined (or detached) before it is destroyed or assigned to. Attempting to destroy a joinable thread will terminate the whole process.

Since

C++11

5.922.2 Member Function Documentation

native_handle()

```
native_handle_type std::thread::native_handle ()  [inline]
```

Precondition

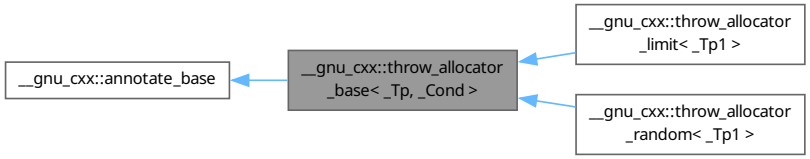
thread is joinable

The documentation for this class was generated from the following file:

- std_thread.h

5.923 __gnu_cxx::throw_allocator_base< _Tp, _Cond > Class Template Reference

`#include <throw_allocator.h>`
Inheritance diagram for __gnu_cxx::throw_allocator_base< _Tp, _Cond >:



Public Types

- typedef const value_type * **const_pointer**
- typedef const value_type & **const_reference**
- typedef std::ptrdiff_t **difference_type**
- typedef value_type * **pointer**
- typedef std::true_type **propagate_on_container_move_assignment**
- typedef value_type & **reference**
- typedef std::size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- const_pointer **address** (const_reference __x) const noexcept
- pointer **address** (reference __x) const noexcept
- pointer **allocate** (size_type __n, const void *__hint=0)
- void **check** (size_type __n)
- map_alloc_type::iterator **check_allocated** (void *p, size_t size)
- void **check_allocated** (pointer __p, size_type __n)
- void **check_constructed** (size_t label)
- map_construct_type::iterator **check_constructed** (void *p)
- template<typename _Up, typename... _Args>
  void **construct** (_Up *__p, _Args &&... __args)

- void **deallocate** (pointer __p, size_type __n)
- template<typename _Up>
  void **destroy** (_Up *__p)
- void **erase** (void *p, size_t size)
- void **erase_construct** (void *p)
- void **insert** (void *p, size_t size)
- void **insert_construct** (void *p)
- size_type **max_size** () const noexcept

Static Public Member Functions

- static void **check** ()
- static size_t **get_label** ()
- static void **set_label** (size_t l)

5.923.1 Detailed Description

template<typename _Tp, typename _Cond>
class __gnu_cxx::throw_allocator_base< _Tp, _Cond >

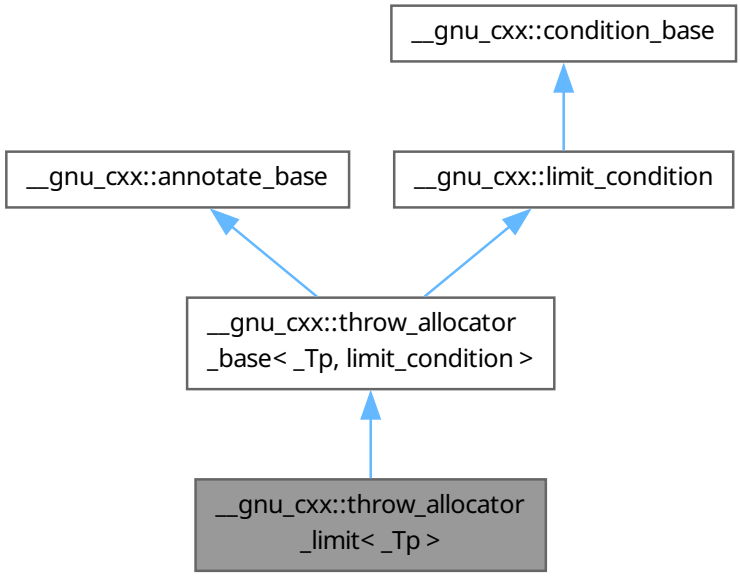
Allocator class with logging and exception generation control. Intended to be used as an allocator_type in templatized code.
Note: Deallocate not allowed to throw.
The documentation for this class was generated from the following file:

- throw_allocator.h

5.924 __gnu_cxx::throw_allocator_limit< _Tp > Struct Template Reference

`#include <throw_allocator.h>`

Inheritance diagram for __gnu_cxx::throw_allocator_limit< _Tp >:



Public Types

- typedef const value_type * **const_pointer**
- typedef const value_type & **const_reference**
- typedef std::ptrdiff_t **difference_type**
- typedef value_type * **pointer**
- typedef std::true_type **propagate_on_container_move_assignment**
- typedef value_type & **reference**
- typedef std::size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **throw_allocator_limit** (const throw_allocator_limit &) noexcept
- template<typename _Tp1>
  throw_allocator_limit (const throw_allocator_limit< _Tp1 > &) noexcept
- const_pointer **address** (const_reference __x) const noexcept
- pointer **address** (reference __x) const noexcept
- pointer **allocate** (size_type __n, const void *__hint=0)
- void **check** (size_type __n)
- map_alloc_type::iterator **check_allocated** (void *p, size_t size)
- void **check_allocated** (pointer __p, size_type __n)
- void **check_constructed** (size_t label)
- map_construct_type::iterator **check_constructed** (void *p)
- void **construct** (_Up *__p, _Args &&... __args)

- void **deallocate** (pointer __p, size_type __n)
- void **destroy** (_Up *__p)
- void **erase** (void *p, size_t size)
- void **erase_construct** (void *p)
- void **insert** (void *p, size_t size)
- void **insert_construct** (void *p)
- size_type **max_size** () const noexcept
- throw_allocator_limit & **operator=** (const throw_allocator_limit &)=default

Static Public Member Functions

- static void **check** ()
- static size_t & **count** ()
- static size_t **get_label** ()
- static size_t & **limit** ()
- static void **set_label** (size_t l)
- static void **set_limit** (const size_t __l)
- static void **throw_conditionally** ()

5.924.1 Detailed Description

template<typename _Tp>
struct __gnu_cxx::throw_allocator_limit< _Tp >

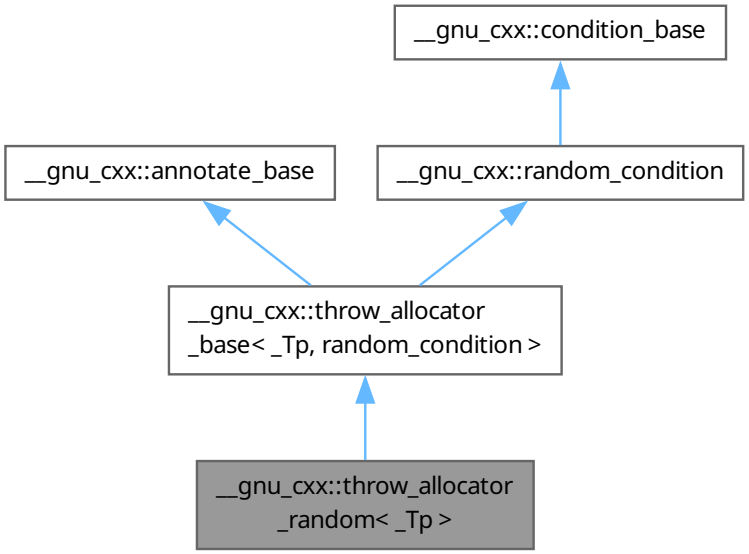
Allocator throwing via limit condition.
The documentation for this struct was generated from the following file:

- throw_allocator.h

5.925 __gnu_cxx::throw_allocator_random< _Tp > Struct Template Reference

`#include <throw_allocator.h>`

Inheritance diagram for __gnu_cxx::throw_allocator_random< _Tp >:



Public Types

- typedef const value_type * **const_pointer**
- typedef const value_type & **const_reference**
- typedef std::ptrdiff_t **difference_type**
- typedef value_type * **pointer**
- typedef std::true_type **propagate_on_container_move_assignment**
- typedef value_type & **reference**
- typedef std::size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **throw_allocator_random** (const throw_allocator_random &) noexcept
- template<typename _Tp1>
  throw_allocator_random (const throw_allocator_random< _Tp1 > &) noexcept
- const_pointer **address** (const_reference __x) const noexcept
- pointer **address** (reference __x) const noexcept
- pointer **allocate** (size_type __n, const void *__hint=0)
- void **check** (size_type __n)
- void **check_allocated** (pointer __p, size_type __n)
- map_alloc_type::iterator **check_allocated** (void *p, size_t size)
- void **check_constructed** (size_t label)
- map_construct_type::iterator **check_constructed** (void *p)
- void **construct** (_Up *__p, _Args &&... __args)
- void **deallocate** (pointer __p, size_type __n)

- void **destroy** (_Up *__p)
- void **erase** (void *p, size_t size)
- void **erase_construct** (void *p)
- void **insert** (void *p, size_t size)
- void **insert_construct** (void *p)
- size_type **max_size** () const noexcept
- throw_allocator_random & **operator=** (const throw_allocator_random &)=default
- void **seed** (unsigned long __s)

Static Public Member Functions

- static void **check** ()
- static size_t **get_label** ()
- static void **set_label** (size_t l)
- static void **set_probability** (double __p)
- static void **throw_conditionally** ()

5.925.1 Detailed Description

template<**typename _Tp**>
struct __gnu_cxx::throw_allocator_random< **_Tp** >

Allocator throwing via random condition.
The documentation for this struct was generated from the following file:

- throw_allocator.h

5.926 __gnu_cxx::throw_value_base< _Cond > Struct Template Reference

`#include <throw_allocator.h>`

Public Types

- typedef _Cond **condition_type**

Public Member Functions

- **throw_value_base** (const std::size_t __i)
- **throw_value_base** (const throw_value_base &__v)
- **throw_value_base** (throw_value_base &&)=default
- throw_value_base & **operator++** ()
- throw_value_base & **operator=** (const throw_value_base &__v)
- throw_value_base & **operator=** (throw_value_base &&)=default

Public Attributes

- std::size_t **_M_i**

5.926.1 Detailed Description

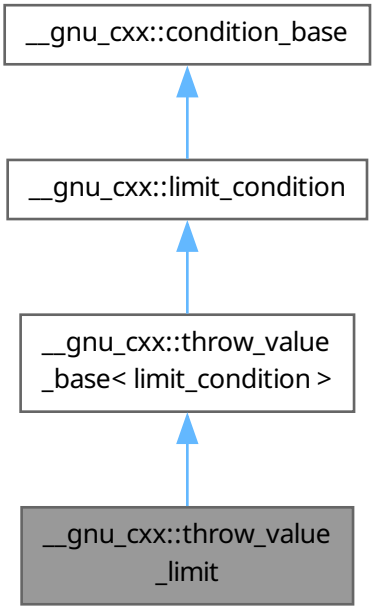
template<**typename _Cond**>
struct __gnu_cxx::throw_value_base< **_Cond** >

Class with exception generation control. Intended to be used as a value_type in templatized code.
Note: Destructor not allowed to throw.
The documentation for this struct was generated from the following file:

- throw_allocator.h

5.927 __gnu_cxx::throw_value_limit Struct Reference

`#include <throw_allocator.h>`
Inheritance diagram for __gnu_cxx::throw_value_limit:



Public Types

- typedef throw_value_base< limit_condition > **base_type**
- typedef limit_condition **condition_type**

Public Member Functions

- **throw_value_limit** (const std::size_t __i)
- **throw_value_limit** (const throw_value_limit &__other)
- **throw_value_limit** (throw_value_limit &&)=default
- throw_value_base & **operator++** ()
- throw_value_base & **operator++** ()
- throw_value_limit & **operator=** (const throw_value_limit &__other)
- throw_value_limit & **operator=** (throw_value_limit &&)=default

Static Public Member Functions

- static size_t & **count** ()
- static size_t & **limit** ()
- static void **set_limit** (const size_t __l)
- static void **throw_conditionally** ()

Public Attributes

- std::size_t **_M_i**
- std::size_t **_M_i**

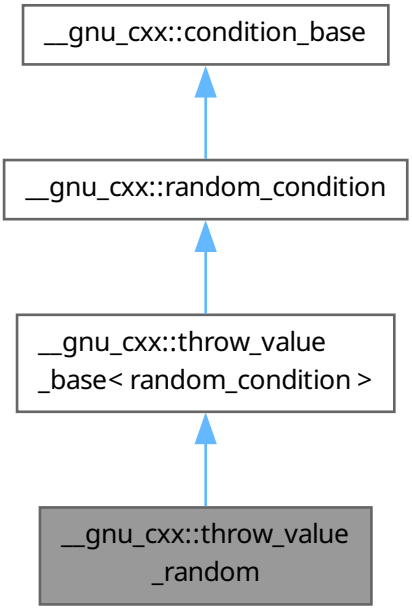
5.927.1 Detailed Description

Type throwing via limit condition.
The documentation for this struct was generated from the following file:

- throw_allocator.h

5.928 __gnu_cxx::throw_value_random Struct Reference

`#include <throw_allocator.h>`
Inheritance diagram for __gnu_cxx::throw_value_random:



Public Types

- typedef throw_value_base< random_condition > **base_type**
- typedef random_condition **condition_type**

Public Member Functions

- **throw_value_random** (const std::size_t __i)
- **throw_value_random** (const throw_value_random &__other)
- **throw_value_random** (throw_value_random &&)=default
- throw_value_base & **operator++** ()

- throw_value_base & **operator++** ()
- throw_value_random & **operator=** (const throw_value_random &__other)
- throw_value_random & **operator=** (throw_value_random &&)=default
- void **seed** (unsigned long __s)

Static Public Member Functions

- static void **set_probability** (double __p)
- static void **throw_conditionally** ()

Public Attributes

- std::size_t **_M_i**
- std::size_t **_M_i**

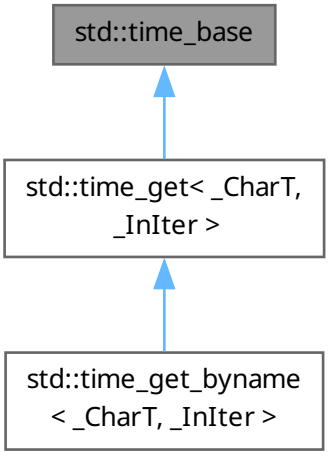
5.928.1 Detailed Description

Type throwing via random condition.
The documentation for this struct was generated from the following file:

- throw_allocator.h

5.929 std::time_base Class Reference

`#include <locale_facets_nonio.h>`
Inheritance diagram for std::time_base:



Public Types

- enum **dateorder** {
  no_order , **dmy** , **mdy** , **ymd** ,
  ydm }

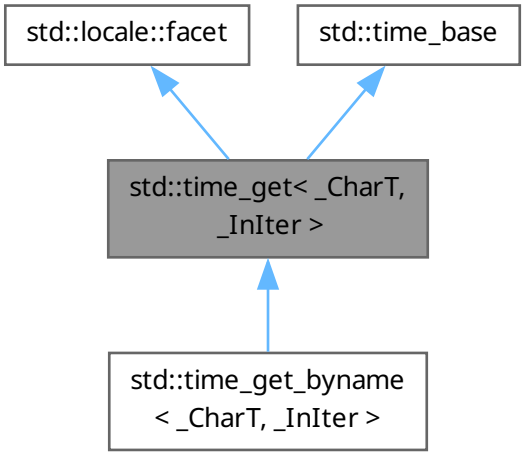
5.929.1 Detailed Description

Time format ordering data.
This class provides an enum representing different orderings of time: day, month, and year.
The documentation for this class was generated from the following file:

- locale_facets_nonio.h

5.930 std::time_get< _CharT, _InIter > Class Template Reference

`#include <locale_facets_nonio.h>`
Inheritance diagram for std::time_get< _CharT, _InIter >:



Public Types

- enum **dateorder** {
  no_order , **dmy** , **mdy** , **ymd** ,
  ydm }

- typedef _CharT char_type
- typedef _InIter iter_type

Public Member Functions

- time_get (size_t __refs=0)
- dateorder date_order () const
- iter_type get (iter_type __s, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm, char __format, char __modifier=0) const
- iter_type get (iter_type __s, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm, const char_type *__fmt, const char_type *__fmtend) const
- iter_type get_date (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const

- iter_type get_monthname (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const
- iter_type get_time (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const
- iter_type get_weekday (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const
- iter_type get_year (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const

Static Public Attributes

- static locale::id id

Protected Member Functions

- virtual ~time_get ()
- iter_type **_M_extract_name** (iter_type __beg, iter_type __end, int &__member, const _CharT **__names, size_t __indexlen, ios_base &__io, ios_base::iostate &__err) const
- iter_type **_M_extract_num** (iter_type __beg, iter_type __end, int &__member, int __min, int __max, size_t __len, ios_base &__io, ios_base::iostate &__err) const
- iter_type **_M_extract_via_format** (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm, const _CharT *__format) const
- iter_type **_M_extract_via_format** (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm, const _CharT *__format, __time_get_state &__state) const
- iter_type **_M_extract_wday_or_month** (iter_type __beg, iter_type __end, int &__member, const _CharT **__names, size_t __indexlen, ios_base &__io, ios_base::iostate &__err) const
- virtual dateorder do_date_order () const
- iter_type do_get (iter_type __s, iter_type __end, ios_base &__f, ios_base::iostate &__err, tm *__tm, char __format, char __modifier) const
- virtual iter_type do_get_date (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const
- virtual iter_type do_get_monthname (iter_type __beg, iter_type __end, ios_base &, ios_base::iostate &__err, tm *__tm) const
- virtual iter_type do_get_time (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const
- virtual iter_type do_get_weekday (iter_type __beg, iter_type __end, ios_base &, ios_base::iostate &__err, tm *__tm) const
- virtual iter_type do_get_year (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

5.930.1 Detailed Description

template<typename _CharT, typename _InIter>
class std::time_get< _CharT, _InIter >

Primary class template time_get.

This facet encapsulates the code to parse and return a date or time from a string. It is used by the istream numeric extraction operators.
The time_get template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the time_get facet.

5.930.2 Member Typedef Documentation

char_type

```
template<typename _CharT, typename _InIter>
typedef _CharT std::time_get< _CharT, _InIter >::char_type
```
Public typedefs.

iter_type

```
template<typename _CharT, typename _InIter>
typedef _InIter std::time_get< _CharT, _InIter >::iter_type
```
Public typedefs.

5.930.3 Constructor & Destructor Documentation

time_get()

```
template<typename _CharT, typename _InIter>
std::time_get< _CharT, _InIter >::time_get (
            size_t __refs = 0)  [inline], [explicit]
```
Constructor performs initialization.
This is the constructor provided by the standard.

Parameters

__refs	Passed to the base facet class.

References std::locale::facet::facet().

~time_get()

```
template<typename _CharT, typename _InIter>
virtual std::time_get< _CharT, _InIter >::~time_get ()  [inline], [protected], [virtual]
```
Destructor.

5.930.4 Member Function Documentation

date_order()

```
template<typename _CharT, typename _InIter>
dateorder std::time_get< _CharT, _InIter >::date_order () const  [inline]
```
Return preferred order of month, day, and year.
This function returns an enum from time_base::dateorder giving the preferred ordering if the format *x* given to time_put::put() only uses month, day, and year. If the format *x* for the associated locale uses other fields, this function returns time_base::dateorder::noorder.
NOTE: The library always returns noorder at the moment.

Returns

A member of time_base::dateorder.

References do_date_order().

do_date_order()

```
template<typename _CharT, typename _InIter>
_GLIBCXX_END_NAMESPACE_LDBL_OR_CXX11 time_base::dateorder std::time_get< _CharT, _InIter >::do_↩
date_order () const  [protected], [virtual]
```

Return preferred order of month, day, and year.
This function returns an enum from time_base::dateorder giving the preferred ordering if the format *x* given to time_↩ put::put() only uses month, day, and year. This function is a hook for derived classes to change the value returned.

Returns

A member of time_base::dateorder.

Referenced by date_order().

do_get()

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::do_get (
            iter_type __s,
            iter_type __end,
            ios_base & __f,
            ios_base::iostate & __err,
            tm * __tm,
            char __format,
            char __modifier) const  [inline], [protected]
```

Parse input string according to format.
This function parses the string according to the provided format and optional modifier. This function is a hook for derived classes to change the value returned.

See also

get() for more details.

Parameters

__s	Start of string to parse.
__end	End of string to parse.
__f	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.
__format	Format specifier.
__modifier	Format modifier.

Returns

Iterator to first char not parsed.

References std::ios_base::_M_getloc(), std::ios_base::eofbit, std::ios_base::goodbit, std::use_facet(), and std::__ctype_abstract_base< _
Referenced by get(), and get().

do_get_date()

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::do_get_date (
```

```
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            tm * __tm) const  [protected], [virtual]
```

Parse input date string.
This function parses a date according to the format *X* and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

get_date() for details.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond date string.

References std::ios_base::_M_getloc(), std::ios_base::eofbit, and std::use_facet().
Referenced by get_date().

do_get_monthname()

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::do_get_monthname (
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            tm * __tm) const  [protected], [virtual]
```

Parse input month string.
This function parses a month name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

get_monthname() for details.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond month name.

References std::ios_base::_M_getloc(), std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, and std::use_facet().
Referenced by get_monthname().

do_get_time()

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::do_get_time (
             iter_type __beg,
             iter_type __end,
             ios_base & __io,
             ios_base::iostate & __err,
             tm * __tm) const  [protected], [virtual]
```

Parse input time string.
This function parses a time according to the format *x* and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

get_time() for details.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond time string.

References std::ios_base::_M_getloc(), std::ios_base::eofbit, and std::use_facet().
Referenced by get_time().

do_get_weekday()

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::do_get_weekday (
             iter_type __beg,
             iter_type __end,
             ios_base & __io,
             ios_base::iostate & __err,
             tm * __tm) const  [protected], [virtual]
```

Parse input weekday string.
This function parses a weekday name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

get_weekday() for details.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond weekday name.

References std::ios_base::_M_getloc(), std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, and std::use_facet().
Referenced by get_weekday().

do_get_year()

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::do_get_year (
             iter_type __beg,
             iter_type __end,
             ios_base & __io,
             ios_base::iostate & __err,
             tm * __tm) const  [protected], [virtual]
```

Parse input year string.
This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

get_year() for details.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond year.

References std::ios_base::_M_getloc(), std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::__ctype_abstract_base< _
and std::use_facet().
Referenced by get_year().

get() [1/2]

```
template<typename _CharT, typename _InIter>
iter_type std::time_get< _CharT, _InIter >::get (
```

```
            iter_type __s,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            tm * __tm,
            char __format,
            char __modifier = 0) const  [inline]
```

Parse input string according to format.
This function calls time_get::do_get with the provided parameters.

See also

do_get() and get().

Parameters

__s	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.
__format	Format specifier.
__modifier	Format modifier.

Returns

Iterator to first char not parsed.

References do_get().

get() [2/2]

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::get (
            iter_type __s,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            tm * __tm,
            const char_type * __fmt,
            const char_type * __fmtend) const  [inline]
```

Parse input string according to format.
This function parses the input string according to a provided format string. It does the inverse of time_put::put. The format string follows the format specified for strftime(3)/strptime(3). The actual parsing is done by time_get::do_get.

Parameters

__s	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.
__fmt	Start of the format string.

__fmtend	End of the format string.

Returns

Iterator to first char not parsed.

References std::ios_base::_M_getloc(), do_get(), std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::__ctype_abstract_base< _CharT >::is(), std::__ctype_abstract_base< _CharT >::narrow(), std::__ctype_abstract_base< _CharT > std::__ctype_abstract_base< _CharT >::toupper(), and std::use_facet().

get_date()

```
template<typename _CharT, typename _InIter>
iter_type std::time_get< _CharT, _InIter >::get_date (
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            tm * __tm) const  [inline]
```

Parse input date string.
This function parses a date according to the format *x* and puts the results into a user-supplied struct tm. The result is returned by calling time_get::do_get_date().
If there is a valid date string according to format *x*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the date string. If an error occurs before the end, err |= ios_base::failbit. If parsing reads all the characters, err |= ios_base::eofbit.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond date string.

References do_get_date().

get_monthname()

```
template<typename _CharT, typename _InIter>
iter_type std::time_get< _CharT, _InIter >::get_monthname (
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            tm * __tm) const  [inline]
```

Parse input month string.
This function parses a month name and puts the results into a user-supplied struct tm. The result is returned by calling time_get::do_get_monthname().

Parsing starts by parsing an abbreviated month name. If a valid abbreviation is followed by a character that would lead to the full month name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.
If an error occurs before the end, err |= ios_base::failbit. If parsing reads all the characters, err |= ios_base::eofbit.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond month name.

References do_get_monthname().

get_time()

```
template<typename _CharT, typename _InIter>
iter_type std::time_get< _CharT, _InIter >::get_time (
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            tm * __tm) const  [inline]
```

Parse input time string.
This function parses a time according to the format *X* and puts the results into a user-supplied struct tm. The result is returned by calling time_get::do_get_time().
If there is a valid time string according to format *X*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the time string. If an error occurs before the end, err |= ios_base::failbit. If parsing reads all the characters, err |= ios_base::eofbit.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond time string.

References do_get_time().

get_weekday()

```
template<typename _CharT, typename _InIter>
iter_type std::time_get< _CharT, _InIter >::get_weekday (
```

```
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            tm * __tm) const  [inline]
```

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct tm. The result is returned by calling time_get::do_get_weekday().

Parsing starts by parsing an abbreviated weekday name. If a valid abbreviation is followed by a character that would lead to the full weekday name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, err |= ios_base::failbit. If parsing reads all the characters, err |= ios_base::eofbit.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond weekday name.

References do_get_weekday().

get_year()

```
template<typename _CharT, typename _InIter>
iter_type std::time_get< _CharT, _InIter >::get_year (
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            tm * __tm) const  [inline]
```

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. The result is returned by calling time_get::do_get_year().

4 consecutive digits are interpreted as a full year. If there are exactly 2 consecutive digits, the library interprets this as the number of years since 1900.

If an error occurs before the end, err |= ios_base::failbit. If parsing reads all the characters, err |= ios_base::eofbit.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond year.

References do_get_year().

5.930.5 Member Data Documentation

id

```
template<typename _CharT, typename _InIter>
locale::id std::time_get< _CharT, _InIter >::id  [static]
```

Numpunct facet id.

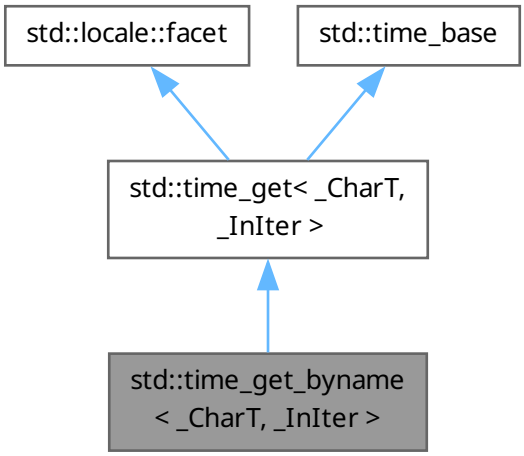
The documentation for this class was generated from the following files:

- locale_facets_nonio.h
- locale_facets_nonio.tcc

5.931 std::time_get_byname< _CharT, _InIter > Class Template Reference

`#include <locale_facets_nonio.h>`

Inheritance diagram for std::time_get_byname< _CharT, _InIter >:



Public Types

- typedef _CharT **char_type**
- enum **dateorder** { **no_order** , **dmy** , **mdy** , **ymd** , **ydm** }
- typedef _InIter **iter_type**

Public Member Functions

- **time_get_byname** (const char *, size_t __refs=0)

- **time_get_byname** (const string &__s, size_t __refs=0)
- dateorder date_order () const
- iter_type get (iter_type __s, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm, char __format, char __modifier=0) const
- iter_type get (iter_type __s, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm, const char_type *__fmt, const char_type *__fmtend) const
- iter_type get_date (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const
- iter_type get_monthname (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const
- iter_type get_time (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const
- iter_type get_weekday (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const
- iter_type get_year (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const

Static Public Attributes

- static locale::id id

Protected Member Functions

- iter_type **_M_extract_name** (iter_type __beg, iter_type __end, int &__member, const _CharT **__names, size_t __indexlen, ios_base &__io, ios_base::iostate &__err) const
- iter_type **_M_extract_num** (iter_type __beg, iter_type __end, int &__member, int __min, int __max, size_t __len, ios_base &__io, ios_base::iostate &__err) const
- iter_type **_M_extract_via_format** (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm, const _CharT *__format) const
- iter_type **_M_extract_via_format** (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm, const _CharT *__format, __time_get_state &__state) const
- iter_type **_M_extract_wday_or_month** (iter_type __beg, iter_type __end, int &__member, const _CharT **__names, size_t __indexlen, ios_base &__io, ios_base::iostate &__err) const
- virtual dateorder do_date_order () const
- iter_type do_get (iter_type __s, iter_type __end, ios_base &__f, ios_base::iostate &__err, tm *__tm, char __format, char __modifier) const
- virtual iter_type do_get_date (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const
- virtual iter_type do_get_monthname (iter_type __beg, iter_type __end, ios_base &, ios_base::iostate &__err, tm *__tm) const
- virtual iter_type do_get_time (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const
- virtual iter_type do_get_weekday (iter_type __beg, iter_type __end, ios_base &, ios_base::iostate &__err, tm *__tm) const
- virtual iter_type do_get_year (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

5.931.1 Detailed Description

template<typename _CharT, typename _InIter>
class std::time_get_byname< _CharT, _InIter >

class time_get_byname [22.2.5.2].

5.931.2 Member Function Documentation

date_order()

```
template<typename _CharT, typename _InIter>
dateorder std::time_get< _CharT, _InIter >::date_order () const  [inline], [inherited]
```

Return preferred order of month, day, and year.
This function returns an enum from time_base::dateorder giving the preferred ordering if the format *x* given to time_↩put::put() only uses month, day, and year. If the format *x* for the associated locale uses other fields, this function returns time_base::dateorder::noorder.
NOTE: The library always returns noorder at the moment.

Returns

A member of time_base::dateorder.

References do_date_order().

do_date_order()

```
template<typename _CharT, typename _InIter>
_GLIBCXX_END_NAMESPACE_LDBL_OR_CXX11 time_base::dateorder std::time_get< _CharT, _InIter >::do_↩
date_order () const  [protected], [virtual], [inherited]
```

Return preferred order of month, day, and year.
This function returns an enum from time_base::dateorder giving the preferred ordering if the format *x* given to time_↩put::put() only uses month, day, and year. This function is a hook for derived classes to change the value returned.

Returns

A member of time_base::dateorder.

Referenced by date_order().

do_get()

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::do_get (
             iter_type __s,
             iter_type __end,
             ios_base & __f,
             ios_base::iostate & __err,
             tm * __tm,
             char __format,
             char __modifier) const  [inline], [protected], [inherited]
```

Parse input string according to format.
This function parses the string according to the provided format and optional modifier. This function is a hook for derived classes to change the value returned.

See also

get() for more details.

Parameters

__s	Start of string to parse.
__end	End of string to parse.
__f	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.
__format	Format specifier.
__modifier	Format modifier.

Returns

Iterator to first char not parsed.

References std::ios_base::_M_getloc(), std::ios_base::eofbit, std::ios_base::goodbit, std::use_facet(), and std::__ctype_abstract_base< _
Referenced by get(), and get().

do_get_date()

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::do_get_date (
             iter_type __beg,
             iter_type __end,
             ios_base & __io,
             ios_base::iostate & __err,
             tm * __tm) const  [protected], [virtual], [inherited]
```

Parse input date string.
This function parses a date according to the format *X* and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

get_date() for details.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond date string.

References std::ios_base::_M_getloc(), std::ios_base::eofbit, and std::use_facet().
Referenced by get_date().

do_get_monthname()

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::do_get_monthname (
```

```
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            tm * __tm) const  [protected], [virtual], [inherited]
```

Parse input month string.
This function parses a month name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

get_monthname() for details.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond month name.

References std::ios_base::_M_getloc(), std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, and std::use_facet().
Referenced by get_monthname().

do_get_time()

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::do_get_time (
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            tm * __tm) const  [protected], [virtual], [inherited]
```

Parse input time string.
This function parses a time according to the format *x* and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

get_time() for details.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond time string.

References std::ios_base::_M_getloc(), std::ios_base::eofbit, and std::use_facet().
Referenced by get_time().

do_get_weekday()

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::do_get_weekday (
             iter_type __beg,
             iter_type __end,
             ios_base & __io,
             ios_base::iostate & __err,
             tm * __tm) const  [protected], [virtual], [inherited]
```

Parse input weekday string.
This function parses a weekday name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

get_weekday() for details.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond weekday name.

References std::ios_base::_M_getloc(), std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, and std::use_facet().
Referenced by get_weekday().

do_get_year()

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::do_get_year (
             iter_type __beg,
             iter_type __end,
             ios_base & __io,
             ios_base::iostate & __err,
             tm * __tm) const  [protected], [virtual], [inherited]
```

Parse input year string.
This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

get_year() for details.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond year.

References std::ios_base::_M_getloc(), std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::__ctype_abstract_base< _
and std::use_facet().
Referenced by get_year().

get() [1/2]

```
template<typename _CharT, typename _InIter>
iter_type std::time_get< _CharT, _InIter >::get (
             iter_type __s,
             iter_type __end,
             ios_base & __io,
             ios_base::iostate & __err,
             tm * __tm,
             char __format,
             char __modifier = 0) const  [inline], [inherited]
```

Parse input string according to format.
This function calls time_get::do_get with the provided parameters.

See also

do_get() and get().

Parameters

__s	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.
__format	Format specifier.
__modifier	Format modifier.

Returns

Iterator to first char not parsed.

References do_get().

get() [2/2]

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::get (
            iter_type __s,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            tm * __tm,
            const char_type * __fmt,
            const char_type * __fmtend) const  [inline], [inherited]
```

Parse input string according to format.

This function parses the input string according to a provided format string. It does the inverse of time_put::put. The format string follows the format specified for strftime(3)/strptime(3). The actual parsing is done by time_get::do_get.

Parameters

__s	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.
__fmt	Start of the format string.
__fmtend	End of the format string.

Returns

Iterator to first char not parsed.

References std::ios_base::_M_getloc(), do_get(), std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::__ctype_abstract_base< _CharT >::is(), std::__ctype_abstract_base< _CharT >::narrow(), std::__ctype_abstract_base< _CharT > std::__ctype_abstract_base< _CharT >::toupper(), and std::use_facet().

get_date()

```
template<typename _CharT, typename _InIter>
iter_type std::time_get< _CharT, _InIter >::get_date (
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            tm * __tm) const  [inline], [inherited]
```

Parse input date string.

This function parses a date according to the format *x* and puts the results into a user-supplied struct tm. The result is returned by calling time_get::do_get_date().

If there is a valid date string according to format *x*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the date string. If an error occurs before the end, err |= ios_base::failbit. If parsing reads all the characters, err |= ios_base::eofbit.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.

Parameters

__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond date string.

References do_get_date().

get_monthname()

```
template<typename _CharT, typename _InIter>
iter_type std::time_get< _CharT, _InIter >::get_monthname (
             iter_type __beg,
             iter_type __end,
             ios_base & __io,
             ios_base::iostate & __err,
             tm * __tm) const  [inline], [inherited]
```

Parse input month string.
This function parses a month name and puts the results into a user-supplied struct tm. The result is returned by calling time_get::do_get_monthname().
Parsing starts by parsing an abbreviated month name. If a valid abbreviation is followed by a character that would lead to the full month name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.
If an error occurs before the end, err |= ios_base::failbit. If parsing reads all the characters, err |= ios_base::eofbit.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond month name.

References do_get_monthname().

get_time()

```
template<typename _CharT, typename _InIter>
iter_type std::time_get< _CharT, _InIter >::get_time (
             iter_type __beg,
             iter_type __end,
             ios_base & __io,
             ios_base::iostate & __err,
             tm * __tm) const  [inline], [inherited]
```

Parse input time string.

This function parses a time according to the format *X* and puts the results into a user-supplied struct tm. The result is returned by calling time_get::do_get_time().
If there is a valid time string according to format *X*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the time string. If an error occurs before the end, err |= ios_base::failbit. If parsing reads all the characters, err |= ios_base::eofbit.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond time string.

References do_get_time().

get_weekday()

```
template<typename _CharT, typename _InIter>
iter_type std::time_get< _CharT, _InIter >::get_weekday (
              iter_type __beg,
              iter_type __end,
              ios_base & __io,
              ios_base::iostate & __err,
              tm * __tm) const  [inline], [inherited]
```

Parse input weekday string.
This function parses a weekday name and puts the results into a user-supplied struct tm. The result is returned by calling time_get::do_get_weekday().
Parsing starts by parsing an abbreviated weekday name. If a valid abbreviation is followed by a character that would lead to the full weekday name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.
If an error occurs before the end, err |= ios_base::failbit. If parsing reads all the characters, err |= ios_base::eofbit.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond weekday name.

References do_get_weekday().

get_year()

```
template<typename _CharT, typename _InIter>
iter_type std::time_get< _CharT, _InIter >::get_year (
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            tm * __tm) const  [inline], [inherited]
```

Parse input year string.
This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. The result is returned by calling time_get::do_get_year().
4 consecutive digits are interpreted as a full year. If there are exactly 2 consecutive digits, the library interprets this as the number of years since 1900.
If an error occurs before the end, err |= ios_base::failbit. If parsing reads all the characters, err |= ios_base::eofbit.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond year.

References do_get_year().

5.931.3 Member Data Documentation

id

```
template<typename _CharT, typename _InIter>
locale::id std::time_get< _CharT, _InIter >::id  [static], [inherited]
```

Numpunct facet id.
The documentation for this class was generated from the following file:

- locale_facets_nonio.h

5.932 std::chrono::time_point< _Clock, _Dur > Class Template Reference

Public Types

- typedef _Clock **clock**
- typedef _Dur **duration**
- typedef duration::period **period**
- typedef duration::rep **rep**

Public Member Functions

- constexpr **time_point** (const duration &__dur)
- template<typename _Dur2, typename = _Require<is_convertible<_Dur2, _Dur>>>
  constexpr **time_point** (const time_point< clock, _Dur2 > &__t)

- constexpr time_point & **operator++** ()
- constexpr time_point **operator++** (int)
- constexpr time_point & **operator+=** (const duration &__dur)
- constexpr time_point & **operator--** ()
- constexpr time_point **operator--** (int)
- constexpr time_point & **operator-=** (const duration &__dur)
- constexpr duration **time_since_epoch** () const

Static Public Member Functions

- static constexpr time_point **max** () noexcept
- static constexpr time_point **min** () noexcept

Related Symbols

(Note that these are not member symbols.)

- template<typename _Clock, typename _Dur1, typename _Rep2, typename _Period2>
  constexpr time_point< _Clock, typename common_type< _Dur1, duration< _Rep2, _Period2 > >::type > operator+ (const time_point< _Clock, _Dur1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)

5.932.1 Detailed Description

template<typename _Clock, typename _Dur>
class std::chrono::time_point< _Clock, _Dur >

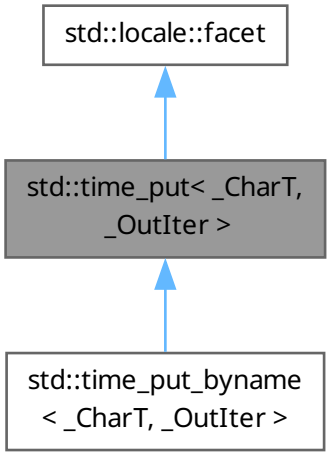
`chrono::time_point` represents a point in time as measured by a clock
The documentation for this class was generated from the following file:

- chrono.h

5.933 std::time_put< _CharT, _OutIter > Class Template Reference

`#include <locale_facets_nonio.h>`

Inheritance diagram for std::time_put< _CharT, _OutIter >:



Public Types

- typedef _CharT char_type
- typedef _OutIter iter_type

Public Member Functions

- time_put (size_t __refs=0)
- iter_type put (iter_type __s, ios_base &__io, char_type __fill, const tm *__tm, char __format, char __mod=0) const
- iter_type put (iter_type __s, ios_base &__io, char_type __fill, const tm *__tm, const _CharT *__beg, const _CharT *__end) const

Static Public Attributes

- static locale::id id

Protected Member Functions

- virtual ~time_put ()
- virtual iter_type do_put (iter_type __s, ios_base &__io, char_type __fill, const tm *__tm, char __format, char __mod) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()

- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

5.933.1 Detailed Description

template<typename _CharT, typename _OutIter>
class std::time_put< _CharT, _OutIter >

Primary class template time_put.
This facet encapsulates the code to format and output dates and times according to formats used by strftime().
The time_put template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the time_put facet.

5.933.2 Member Typedef Documentation

char_type

```
template<typename _CharT, typename _OutIter>
typedef _CharT std::time_put< _CharT, _OutIter >::char_type
```

Public typedefs.

iter_type

```
template<typename _CharT, typename _OutIter>
typedef _OutIter std::time_put< _CharT, _OutIter >::iter_type
```

Public typedefs.

5.933.3 Constructor & Destructor Documentation

time_put()

```
template<typename _CharT, typename _OutIter>
std::time_put< _CharT, _OutIter >::time_put (
            size_t __refs = 0)  [inline], [explicit]
```

Constructor performs initialization.
This is the constructor provided by the standard.

Parameters

__refs	Passed to the base facet class.

References std::locale::facet::facet().

~time_put()

```
template<typename _CharT, typename _OutIter>
virtual std::time_put< _CharT, _OutIter >::~time_put ()  [inline], [protected], [virtual]
```

Destructor.

5.933.4 Member Function Documentation

do_put()

```
template<typename _CharT, typename _OutIter>
_OutIter std::time_put< _CharT, _OutIter >::do_put (
```

```
            iter_type __s,
            ios_base & __io,
            char_type __fill,
            const tm * __tm,
            char __format,
            char __mod) const  [protected], [virtual]
```

Format and output a time or date.
This function formats the data in struct tm according to the provided format char and optional modifier. This function is a hook for derived classes to change the value returned.

See also

put() for more details.

Parameters

__s	The stream to write to.
__io	Source of locale.
__fill	char_type to use for padding.
__tm	Struct tm with date and time info to format.
__format	Format char.
__mod	Optional modifier char.

Returns

Iterator after writing.

References std::ios_base::_M_getloc(), std::use_facet(), and std::__ctype_abstract_base< _CharT >::widen().
Referenced by put(), and put().

put() [1/2]

```
template<typename _CharT, typename _OutIter>
iter_type std::time_put< _CharT, _OutIter >::put (
            iter_type __s,
            ios_base & __io,
            char_type __fill,
            const tm * __tm,
            char __format,
            char __mod = 0) const  [inline]
```

Format and output a time or date.
This function formats the data in struct tm according to the provided format char and optional modifier. The format and modifier are interpreted as by strftime(). It does so by returning time_put::do_put().

Parameters

__s	The stream to write to.
__io	Source of locale.
__fill	char_type to use for padding.
__tm	Struct tm with date and time info to format.
__format	Format char.
__mod	Optional modifier char.

Returns

Iterator after writing.

References do_put().

put() [2/2]

```
template<typename _CharT, typename _OutIter>
_OutIter std::time_put< _CharT, _OutIter >::put (
            iter_type __s,
            ios_base & __io,
            char_type __fill,
            const tm * __tm,
            const _CharT * __beg,
            const _CharT * __end) const
```

Format and output a time or date.
This function formats the data in struct tm according to the provided format string. The format string is interpreted as by strftime().

Parameters

__s	The stream to write to.
__io	Source of locale.
__fill	char_type to use for padding.
__tm	Struct tm with date and time info to format.
__beg	Start of format string.
__end	End of format string.

Returns

Iterator after writing.

References std::ios_base::_M_getloc(), do_put(), std::__ctype_abstract_base< _CharT >::narrow(), and std::use_facet().

5.933.5 Member Data Documentation

id

```
template<typename _CharT, typename _OutIter>
locale::id std::time_put< _CharT, _OutIter >::id  [static]
```

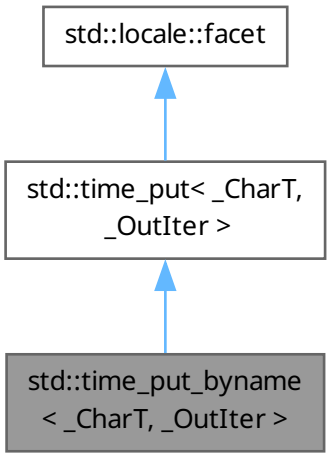
Numpunct facet id.
The documentation for this class was generated from the following files:

- locale_facets_nonio.h
- locale_facets_nonio.tcc

5.934 std::time_put_byname< _CharT, _OutIter > Class Template Reference

```
#include <locale_facets_nonio.h>
```

Inheritance diagram for std::time_put_byname< _CharT, _OutIter >:



Public Types

- typedef _CharT **char_type**
- typedef _OutIter **iter_type**

Public Member Functions

- **time_put_byname** (const char *, size_t __refs=0)
- **time_put_byname** (const string &__s, size_t __refs=0)
- iter_type put (iter_type __s, ios_base &__io, char_type __fill, const tm *__tm, char __format, char __mod=0) const
- iter_type put (iter_type __s, ios_base &__io, char_type __fill, const tm *__tm, const _CharT *__beg, const _CharT *__end) const

Static Public Attributes

- static locale::id id

Protected Member Functions

- virtual iter_type do_put (iter_type __s, ios_base &__io, char_type __fill, const tm *__tm, char __format, char __mod) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

5.934.1 Detailed Description

template<typename _CharT, typename _OutIter>
class std::time_put_byname< _CharT, _OutIter >

class time_put_byname [22.2.5.4].

5.934.2 Member Function Documentation

do_put()

```
template<typename _CharT, typename _OutIter>
_OutIter std::time_put< _CharT, _OutIter >::do_put (
            iter_type __s,
            ios_base & __io,
            char_type __fill,
            const tm * __tm,
            char __format,
            char __mod) const  [protected], [virtual], [inherited]
```

Format and output a time or date.
This function formats the data in struct tm according to the provided format char and optional modifier. This function is a hook for derived classes to change the value returned.

See also

put() for more details.

Parameters

__s	The stream to write to.
__io	Source of locale.
__fill	char_type to use for padding.
__tm	Struct tm with date and time info to format.
__format	Format char.
__mod	Optional modifier char.

Returns

Iterator after writing.

References std::ios_base::_M_getloc(), std::use_facet(), and std::__ctype_abstract_base< _CharT >::widen().
Referenced by put(), and put().

put() [1/2]

```
template<typename _CharT, typename _OutIter>
iter_type std::time_put< _CharT, _OutIter >::put (
            iter_type __s,
            ios_base & __io,
            char_type __fill,
            const tm * __tm,
            char __format,
            char __mod = 0) const  [inline], [inherited]
```

Format and output a time or date.
This function formats the data in struct tm according to the provided format char and optional modifier. The format and modifier are interpreted as by strftime(). It does so by returning time_put::do_put().

Parameters

__s	The stream to write to.
__io	Source of locale.
__fill	char_type to use for padding.
__tm	Struct tm with date and time info to format.
__format	Format char.
__mod	Optional modifier char.

Returns

Iterator after writing.

References do_put().

put() [2/2]

```
template<typename _CharT, typename _OutIter>
_OutIter std::time_put< _CharT, _OutIter >::put (
            iter_type __s,
            ios_base & __io,
            char_type __fill,
            const tm * __tm,
            const _CharT * __beg,
            const _CharT * __end) const  [inherited]
```

Format and output a time or date.
This function formats the data in struct tm according to the provided format string. The format string is interpreted as by strftime().

Parameters

__s	The stream to write to.
__io	Source of locale.
__fill	char_type to use for padding.
__tm	Struct tm with date and time info to format.
__beg	Start of format string.
__end	End of format string.

Returns

Iterator after writing.

References std::ios_base::_M_getloc(), do_put(), std::__ctype_abstract_base< _CharT >::narrow(), and std::use_facet().

5.934.3 Member Data Documentation

id

```
template<typename _CharT, typename _OutIter>
locale::id std::time_put< _CharT, _OutIter >::id  [static], [inherited]
```

Numpunct facet id.
The documentation for this class was generated from the following file:

- locale_facets_nonio.h

5.935 std::timed_mutex Class Reference

`#include <mutex>`

Public Types

- typedef __native_type * **native_handle_type**

Public Member Functions

- **timed_mutex** (const timed_mutex &)=delete
- void **lock** ()
- native_handle_type **native_handle** () noexcept
- timed_mutex & **operator=** (const timed_mutex &)=delete
- bool **try_lock** () noexcept
- template<class _Rep, class _Period>
  bool **try_lock_for** (const chrono::duration< _Rep, _Period > &__rtime)
- template<class _Clock, class _Duration>
  bool **try_lock_until** (const chrono::time_point< _Clock, _Duration > &__atime)
- void **unlock** ()

Friends

- class **__timed_mutex_impl**< **timed_mutex** >

5.935.1 Detailed Description

The standard timed mutex type.
A non-recursive mutex that supports a timeout when trying to acquire the lock.

Since

C++11

The documentation for this class was generated from the following file:

- mutex

5.936 std::to_chars_result Struct Reference

`#include <charconv>`

Public Attributes

- errc **ec**
- char * **ptr**

Friends

- bool **operator==** (const to_chars_result &, const to_chars_result &)=default

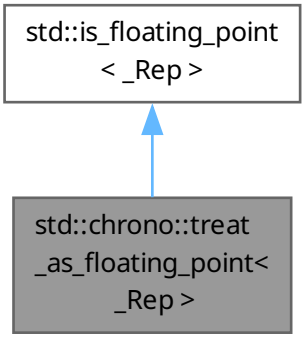
5.936.1 Detailed Description

Result type of std::to_chars.
The documentation for this struct was generated from the following file:

- charconv

5.937 std::chrono::treat_as_floating_point< _Rep > Struct Template Reference

`#include <chrono.h>`
Inheritance diagram for std::chrono::treat_as_floating_point< _Rep >:



5.937.1 Detailed Description

template<typename _Rep>
struct std::chrono::treat_as_floating_point< _Rep >

Trait indicating whether to treat a type as a floating-point type.
The chrono library uses this trait to tell whether a `duration` can represent fractional values of the given precision, or only integral values.
You should specialize this trait for your own numeric types that are used with `duration` and can represent non-integral values.

Since

C++11

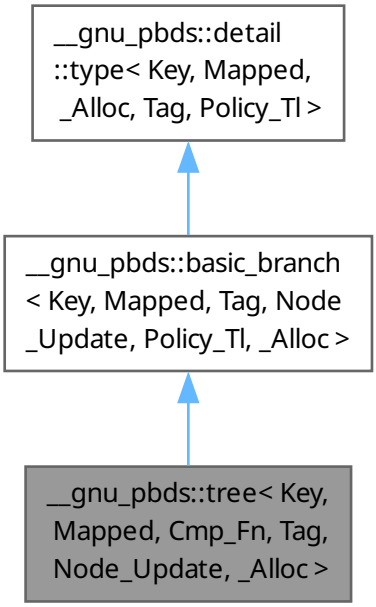
The documentation for this struct was generated from the following file:

- chrono.h

5.938 __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc > Class Template Reference

`#include <assoc_container.hpp>`

Inheritance diagram for __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >:



Public Types

- typedef Cmp_Fn cmp_fn
- typedef Node_Update **node_update**

Public Member Functions

- tree (const cmp_fn &c)
- **tree** (const tree &other)
- template<typename It>
  tree (It first, It last)
- template<typename It>
  tree (It first, It last, const cmp_fn &c)
- tree & **operator=** (const tree &other)
- void **swap** (tree &other)

5.938.1 Detailed Description

template<typename Key, typename Mapped, typename Cmp_Fn = std::less<Key>, typename Tag = rb_tree_tag, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc = std::allocator<char>>
class __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >

A tree-based container.

Template Parameters

Key	Key type.
Mapped	Map type.
Cmp_Fn	Comparison functor.
Tag	Instantiating data structure type, see container_tag.
Node_Update	Updates tree internal-nodes, restores invariants when invalidated. XXX See design::tree-based-containersnode invariants.
_Alloc	Allocator type.

Base tag choices are: ov_tree_tag, rb_tree_tag, splay_tree_tag.
Base is basic_branch.

5.938.2 Member Typedef Documentation

cmp_fn

```
template<typename Key, typename Mapped, typename Cmp_Fn = std::less<Key>, typename Tag = rb_↩
tree_tag, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ >
class Node_Update = null_node_update, typename _Alloc = std::allocator<char>>
typedef Cmp_Fn __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >::cmp_fn
```

Comparison functor type.

5.938.3 Constructor & Destructor Documentation

tree() [1/3]

```
template<typename Key, typename Mapped, typename Cmp_Fn = std::less<Key>, typename Tag = rb_↩
tree_tag, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ >
class Node_Update = null_node_update, typename _Alloc = std::allocator<char>>
__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >::tree (
            const cmp_fn & c)  [inline]
```

Constructor taking some policy objects. r_cmp_fn will be copied by the Cmp_Fn object of the container object.

tree() [2/3]

```
template<typename Key, typename Mapped, typename Cmp_Fn = std::less<Key>, typename Tag = rb_↩
tree_tag, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ >
class Node_Update = null_node_update, typename _Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >::tree (
            It first,
            It last)  [inline]
```

Constructor taking __iterators to a range of value_types. The value_types between first_it and last_it will be inserted into the container object.

tree() [3/3]

```
template<typename Key, typename Mapped, typename Cmp_Fn = std::less<Key>, typename Tag = rb_↩
tree_tag, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ >
class Node_Update = null_node_update, typename _Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >::tree (
            It first,
```

```
It last,
const cmp_fn & c)  [inline]
```

Constructor taking __iterators to a range of value_types and some policy objects The value_types between first_it and last_it will be inserted into the container object. r_cmp_fn will be copied by the cmp_fn object of the container object.
The documentation for this class was generated from the following file:

- assoc_container.hpp

5.939 __gnu_pbds::detail::tree_metadata_helper< Node_Update, _BTp > Struct Template Reference

5.939.1 Detailed Description

template<typename Node_Update, bool _BTp>
struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, _BTp >

Tree metadata helper.
The documentation for this struct was generated from the following file:

- tree_policy/node_metadata_selector.hpp

5.940 __gnu_pbds::detail::tree_metadata_helper< Node_Update, false > Struct Template Reference

`#include <node_metadata_selector.hpp>`

Public Types

- typedef Node_Update::metadata_type **type**

5.940.1 Detailed Description

template<typename Node_Update>
struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, false >

Specialization, false.
The documentation for this struct was generated from the following file:

- tree_policy/node_metadata_selector.hpp

5.941 __gnu_pbds::detail::tree_metadata_helper< Node_Update, true > Struct Template Reference

`#include <node_metadata_selector.hpp>`

Public Types

- typedef null_type **type**

5.941.1 Detailed Description

template<typename Node_Update>
struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, true >

Specialization, true.
The documentation for this struct was generated from the following file:

- tree_policy/node_metadata_selector.hpp

5.942 __gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc > Struct Template Reference

`#include <node_metadata_selector.hpp>`

Public Types

- typedef tree_metadata_helper< __node_u, null_update >::type **type**

5.942.1 Detailed Description

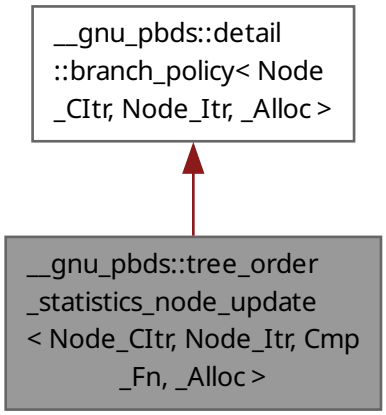
template<typename Key, typename Data, typename Cmp_Fn, template< typename Node_CItr, typename Const_Iterator, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >

Tree node metadata dispatch.
The documentation for this struct was generated from the following file:

- tree_policy/node_metadata_selector.hpp

5.943 __gnu_pbds::tree_order_statistics_node_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc > Class Template Reference

`#include <tree_policy.hpp>`
Inheritance diagram for __gnu_pbds::tree_order_statistics_node_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc >:



Public Types

- typedef _Alloc **allocator_type**
- typedef Cmp_Fn **cmp_fn**
- typedef node_const_iterator::value_type **const_iterator**
- typedef node_iterator::value_type **iterator**
- typedef base_type::key_const_reference **key_const_reference**
- typedef base_type::key_type **key_type**

- typedef size_type **metadata_type**
- typedef Node_CItr **node_const_iterator**
- typedef Node_Itr **node_iterator**
- typedef allocator_type::size_type **size_type**

Public Member Functions

- iterator find_by_order (size_type)
- const_iterator find_by_order (size_type) const
- size_type order_of_key (key_const_reference) const

Protected Member Functions

- void operator() (node_iterator, node_const_iterator) const

5.943.1 Detailed Description

template<typename Node_CItr, typename Node_Itr, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::tree_order_statistics_node_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc >

Functor updating ranks of entrees.

5.943.2 Member Function Documentation

find_by_order() [1/2]

```
template<typename Node_CItr, typename Node_Itr, typename Cmp_Fn, typename _Alloc>
tree_order_statistics_node_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc >::iterator __gnu_pbds::tree_order_statist
Node_CItr, Node_Itr, Cmp_Fn, _Alloc >::find_by_order (
            size_type order)  [inline]
```

Finds an entry by __order. Returns an iterator to the entry with the __order order, or an iterator to the container object's end if order is at least the size of the container object.

find_by_order() [2/2]

```
template<typename Node_CItr, typename Node_Itr, typename Cmp_Fn, typename _Alloc>
tree_order_statistics_node_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc >::const_iterator __gnu_pbds::tree_order_s
Node_CItr, Node_Itr, Cmp_Fn, _Alloc >::find_by_order (
            size_type order) const  [inline]
```

Finds an entry by __order. Returns a const_iterator to the entry with the __order order, or a const_iterator to the container object's end if order is at least the size of the container object.

operator()()

```
template<typename Node_CItr, typename Node_Itr, typename Cmp_Fn, typename _Alloc>
void __gnu_pbds::tree_order_statistics_node_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc >::operator()
(
            node_iterator node_it,
            node_const_iterator end_nd_it) const  [inline], [protected]
```

Updates the rank of a node through a node_iterator node_it; end_nd_it is the end node iterator.

order_of_key()

```
template<typename Node_CItr, typename Node_Itr, typename Cmp_Fn, typename _Alloc>
tree_order_statistics_node_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc >::size_type __gnu_pbds::tree_order_statis
```

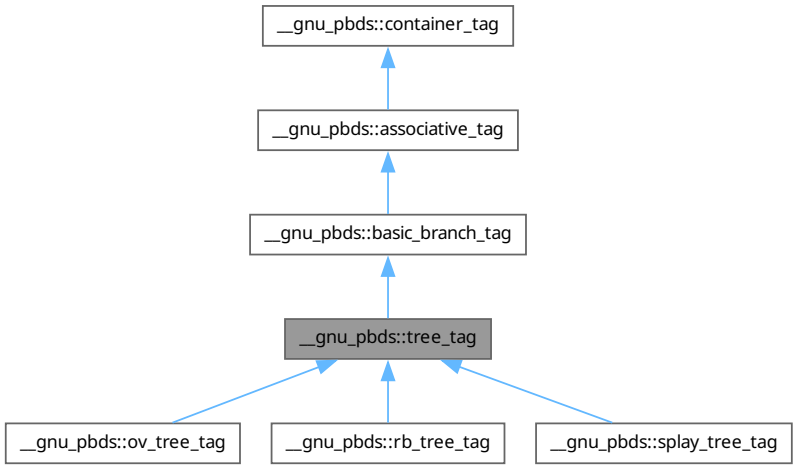
```
Node_CItr, Node_Itr, Cmp_Fn, _Alloc >::order_of_key (
            key_const_reference r_key) const  [inline]
```

Returns the order of a key within a sequence. For exapmle, if r_key is the smallest key, this method will return 0; if r_key is a key between the smallest and next key, this method will return 1; if r_key is a key larger than the largest key, this method will return the size of r_c.
The documentation for this class was generated from the following file:

- tree_policy.hpp

5.944 __gnu_pbds::tree_tag Struct Reference

`#include <tag_and_trait.hpp>`
Inheritance diagram for __gnu_pbds::tree_tag:



5.944.1 Detailed Description

Basic tree structure.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

5.945 __gnu_pbds::detail::tree_traits< Key, Data, Cmp_Fn, Node_Update, Tag, _Alloc > Struct Template Reference

5.945.1 Detailed Description

template<typename Key, typename Data, typename Cmp_Fn, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc > class Node_Update, typename Tag, typename _Alloc> struct __gnu_pbds::detail::tree_traits< Key, Data, Cmp_Fn, Node_Update, Tag, _Alloc >

Tree traits class, primary template.
The documentation for this struct was generated from the following file:

- branch_policy/traits.hpp

5.946 __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc > Struct Template Reference

`#include <traits.hpp>`

Public Types

- typedef tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type **metadata_type**
- typedef ov_tree_node_const_it_< value_type, metadata_type, _Alloc > node_const_iterator
- typedef ov_tree_node_it_< value_type, metadata_type, _Alloc > **node_iterator**
- typedef Node_Update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > **node_update**
- typedef __gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * **null_node_update_pointer**

5.946.1 Detailed Description

template<typename Key, typename Mapped, class Cmp_Fn, template< typename Node_CItr, class Node_Itr, class Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >

Tree traits.

5.946.2 Member Typedef Documentation

node_const_iterator

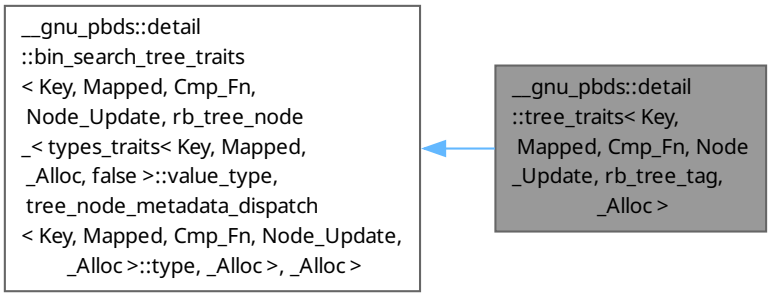
```
template<typename Key, typename Mapped, class Cmp_Fn, template< typename Node_CItr, class Node_Itr, class Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
typedef ov_tree_node_const_it_< value_type, metadata_type, _Alloc> __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >::node_const_iterator
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.
The documentation for this struct was generated from the following file:

- ov_tree_map_/traits.hpp

5.947 __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc > Struct Template Reference

`#include <traits.hpp>`
Inheritance diagram for __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >:



Public Types

- typedef bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc > **const_reverse_iterator**
- typedef rb_tree_node_< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc > **node**
- typedef bin_search_tree_const_node_it_< rb_tree_node_< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc > node_const_iterator
- typedef bin_search_tree_node_it_< rb_tree_node_< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc > **node_iterator**
- typedef Node_Update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > **node_update**
- typedef __gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * **null_node_update_pointer**
- typedef bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc > **point_const_iterator**
- typedef bin_search_tree_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc > **point_iterator**
- typedef bin_search_tree_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc > **reverse_iterator**

5.947.1 Detailed Description

template<typename Key, typename Mapped, typename Cmp_Fn, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >

Specialization.

5.947.2 Member Typedef Documentation

node_const_iterator

```
typedef bin_search_tree_const_node_it_< rb_tree_node_< types_traits< Key, Mapped, _Alloc, false
>::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _↩
Alloc >, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits<
Key, Mapped, Cmp_Fn, Node_Update, rb_tree_node_< types_traits< Key, Mapped, _Alloc, false >↩
::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _↩
Alloc >, _Alloc >::node_const_iterator [inherited]
```

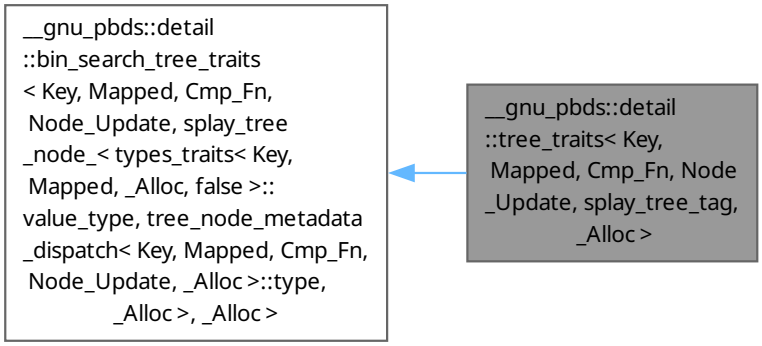
This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.
The documentation for this struct was generated from the following file:

- rb_tree_map_/traits.hpp

5.948 __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc > Struct Template Reference

```
#include <traits.hpp>
```

Inheritance diagram for __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >:



Public Types

- typedef bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc > **const_reverse_iterator**
- typedef splay_tree_node_< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc > **node**
- typedef bin_search_tree_const_node_it_< splay_tree_node_< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc > node_const_iterator
- typedef bin_search_tree_node_it_< splay_tree_node_< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc > **node_iterator**
- typedef Node_Update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > **node_update**
- typedef __gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * **null_node_update_pointer**
- typedef bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc > **point_const_iterator**
- typedef bin_search_tree_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc > **point_iterator**
- typedef bin_search_tree_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc > **reverse_iterator**

5.948.1 Detailed Description

template<typename Key, typename Mapped, typename Cmp_Fn, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >

Specialization.

5.948.2 Member Typedef Documentation

node_const_iterator

```
typedef bin_search_tree_const_node_it_< splay_tree_node_< types_traits< Key, Mapped, _Alloc,
false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >↩
::type, _Alloc >, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits<
Key, Mapped, Cmp_Fn, Node_Update, splay_tree_node_< types_traits< Key, Mapped, _Alloc, false >↩
::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _↩
Alloc >, _Alloc >::node_const_iterator  [inherited]
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.
The documentation for this struct was generated from the following file:

- splay_tree_/traits.hpp

5.949 __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc > Struct Template Reference

```
#include <traits.hpp>
```

Public Types

- typedef tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type **metadata_type**
- typedef ov_tree_node_const_it_< value_type, metadata_type, _Alloc > node_const_iterator
- typedef node_const_iterator **node_iterator**
- typedef Node_Update< node_const_iterator, node_const_iterator, Cmp_Fn, _Alloc > **node_update**
- typedef __gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * **null_node↩ _update_pointer**

5.949.1 Detailed Description

template<typename Key, class Cmp_Fn, template< typename Node_CItr, class Node_Itr, class Cmp_Fn_↩ , typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >

Specialization.

5.949.2 Member Typedef Documentation

node_const_iterator

```
template<typename Key, class Cmp_Fn, template< typename Node_CItr, class Node_Itr, class Cmp_Fn↩
_, typename _Alloc_ > class Node_Update, typename _Alloc>
typedef ov_tree_node_const_it_< value_type, metadata_type, _Alloc> __gnu_pbds::detail::tree_traits<
Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >::node_const_iterator
```

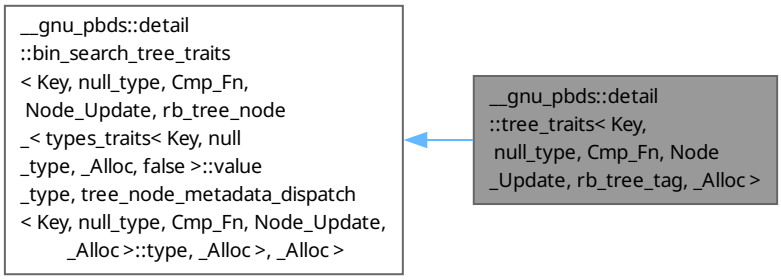
This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.
The documentation for this struct was generated from the following file:

- ov_tree_map_/traits.hpp

5.950 __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc > Struct Template Reference

```
#include <traits.hpp>
```

Inheritance diagram for __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >:



Public Types

- typedef bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc > **const_reverse_iterator**
- typedef rb_tree_node_< types_traits< Key, null_type, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc > **node**
- typedef bin_search_tree_const_node_it_< rb_tree_node_< types_traits< Key, null_type, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc > node_const_iterator
- typedef bin_search_tree_node_it_< rb_tree_node_< types_traits< Key, null_type, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc > **node_iterator**
- typedef Node_Update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > **node_update**
- typedef __gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * **null_node_update_pointer**
- typedef bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc > **point_const_iterator**
- typedef bin_search_tree_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc > **point_iterator**
- typedef bin_search_tree_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc > **reverse_iterator**

5.950.1 Detailed Description

template<typename Key, typename Cmp_Fn, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >

Specialization.

5.950.2 Member Typedef Documentation

node_const_iterator

```
typedef bin_search_tree_const_node_it_< rb_tree_node_< types_traits< Key, null_type, _Alloc,
false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >↩
::type, _Alloc >, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits<
Key, null_type, Cmp_Fn, Node_Update, rb_tree_node_< types_traits< Key, null_type, _Alloc, false
>::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type,
_Alloc >, _Alloc >::node_const_iterator [inherited]
```

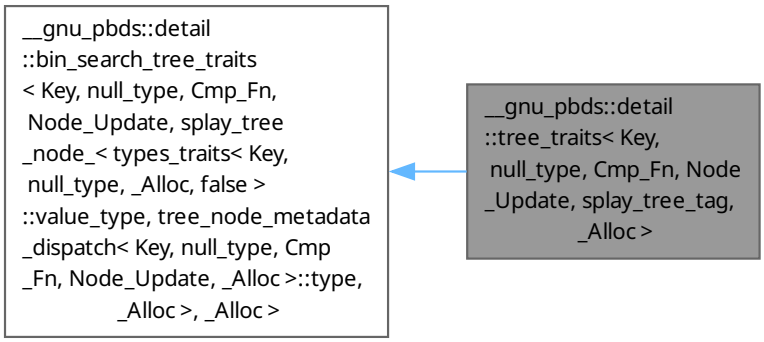
This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.
The documentation for this struct was generated from the following file:

- rb_tree_map_/traits.hpp

5.951 __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc > Struct Template Reference

```
#include <traits.hpp>
```

Inheritance diagram for __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >:



Public Types

- typedef bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc > **const_reverse_iterator**
- typedef splay_tree_node_< types_traits< Key, null_type, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc > **node**
- typedef bin_search_tree_const_node_it_< splay_tree_node_< types_traits< Key, null_type, _Alloc, false >↩ ::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc > node_const_iterator
- typedef bin_search_tree_node_it_< splay_tree_node_< types_traits< Key, null_type, _Alloc, false >::value↩ _type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc > **node_iterator**
- typedef Node_Update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > **node_update**
- typedef __gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * **null_node↩ _update_pointer**

- typedef bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc > **point_const_iterator**
- typedef bin_search_tree_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc > **point_iterator**
- typedef bin_search_tree_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc > **reverse_iterator**

5.951.1 Detailed Description

template<typename Key, class Cmp_Fn, template< typename Node_CItr, class Node_Itr, class Cmp_Fn_↩ , typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >

Specialization.

5.951.2 Member Typedef Documentation

node_const_iterator

```
typedef bin_search_tree_const_node_it_< splay_tree_node_< types_traits< Key, null_type, _Alloc,
false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >↩
::type, _Alloc >, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits<
Key, null_type, Cmp_Fn, Node_Update, splay_tree_node_< types_traits< Key, null_type, _Alloc,
false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >↩
::type, _Alloc >, _Alloc >::node_const_iterator [inherited]
```

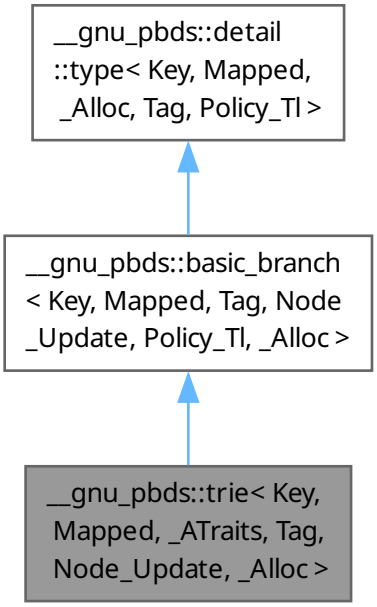
This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.
The documentation for this struct was generated from the following file:

- splay_tree_/traits.hpp

5.952 __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc > Class Template Reference

```
#include <assoc_container.hpp>
```

Inheritance diagram for __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >:



Public Types

- typedef _ATraits access_traits
- typedef Node_Update **node_update**

Public Member Functions

- trie (const access_traits &t)
- **trie** (const trie &other)
- template<typename It>
  trie (It first, It last)
- template<typename It>
  trie (It first, It last, const access_traits &t)
- trie & **operator=** (const trie &other)
- void **swap** (trie &other)

5.952.1 Detailed Description

template<typename Key, typename Mapped, typename _ATraits = typename detail::default_trie_access_↩ traits<Key>::type, typename Tag = pat_trie_tag, template< typename Node_CItr, typename Node_Itr, type-name _ATraits_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc = std↩ ::allocator<char>>
class __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >

A trie-based container.

Template Parameters

Key	Key type.
Mapped	Map type.
_ATraits	Element access traits.
Tag	Instantiating data structure type, see container_tag.
Node_Update	Updates trie internal-nodes, restores invariants when invalidated. XXX See design::tree-based-containersnode invariants.
_Alloc	Allocator type.

Base tag choice is pat_trie_tag.
Base is basic_branch.

5.952.2 Member Typedef Documentation

access_traits

```
template<typename Key, typename Mapped, typename _ATraits = typename detail::default_trie_access↩
_traits<Key>::type, typename Tag = pat_trie_tag, template< typename Node_CItr, typename Node_↩
Itr, typename _ATraits_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc
= std::allocator<char>>
typedef _ATraits __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >::access_↩
traits
```

Element access traits type.

5.952.3 Constructor & Destructor Documentation

trie() [1/3]

```
template<typename Key, typename Mapped, typename _ATraits = typename detail::default_trie_access↩
_traits<Key>::type, typename Tag = pat_trie_tag, template< typename Node_CItr, typename Node_↩
Itr, typename _ATraits_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc
= std::allocator<char>>
__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >::trie (
            const access_traits & t)  [inline]
```

Constructor taking some policy objects. r_access_traits will be copied by the _ATraits object of the container object.

trie() [2/3]

```
template<typename Key, typename Mapped, typename _ATraits = typename detail::default_trie_access↩
_traits<Key>::type, typename Tag = pat_trie_tag, template< typename Node_CItr, typename Node_↩
Itr, typename _ATraits_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc
= std::allocator<char>>
template<typename It>
__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >::trie (
            It first,
            It last)  [inline]
```

Constructor taking __iterators to a range of value_types. The value_types between first_it and last_it will be inserted into the container object.

trie() [3/3]

```
template<typename Key, typename Mapped, typename _ATraits = typename detail::default_trie_access↩
_traits<Key>::type, typename Tag = pat_trie_tag, template< typename Node_CItr, typename Node_↩
Itr, typename _ATraits_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc
```

```
= std::allocator<char>>
template<typename It>
__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >::trie (
             It first,
             It last,
             const access_traits & t)  [inline]
```

Constructor taking __iterators to a range of value_types and some policy objects. The value_types between first_it and last_it will be inserted into the container object.
The documentation for this class was generated from the following file:

- assoc_container.hpp

5.953 __gnu_pbds::detail::trie_metadata_helper< Node_Update, _BTp > Struct Template Reference

5.953.1 Detailed Description

template<typename Node_Update, bool _BTp>
struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, _BTp >

Trie metadata helper.
The documentation for this struct was generated from the following file:

- trie_policy/node_metadata_selector.hpp

5.954 __gnu_pbds::detail::trie_metadata_helper< Node_Update, false > Struct Template Reference

`#include <node_metadata_selector.hpp>`

Public Types

- typedef Node_Update::metadata_type **type**

5.954.1 Detailed Description

template<typename Node_Update>
struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, false >

Specialization, false.
The documentation for this struct was generated from the following file:

- trie_policy/node_metadata_selector.hpp

5.955 __gnu_pbds::detail::trie_metadata_helper< Node_Update, true > Struct Template Reference

`#include <node_metadata_selector.hpp>`

Public Types

- typedef null_type **type**

5.955.1 Detailed Description

template<typename Node_Update>
struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, true >

Specialization, true.
The documentation for this struct was generated from the following file:

- trie_policy/node_metadata_selector.hpp

5.956 __gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc > Struct Template Reference

`#include <node_metadata_selector.hpp>`

Public Types

- typedef trie_metadata_helper< __node_u, null_update >::type **type**

5.956.1 Detailed Description

template<typename Key, typename Data, typename Cmp_Fn, template< typename Node_CItr, typename Const_Iterator, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >

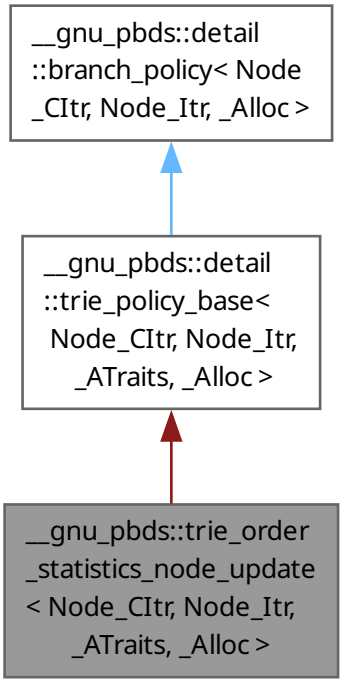
Trie node metadata dispatch.
The documentation for this struct was generated from the following file:

- trie_policy/node_metadata_selector.hpp

5.957 __gnu_pbds::trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc > Class Template Reference

`#include <trie_policy.hpp>`

Inheritance diagram for __gnu_pbds::trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >:



Public Types

- typedef access_traits::const_iterator **a_const_iterator**
- typedef _ATraits **access_traits**
- typedef _Alloc **allocator_type**
- typedef node_const_iterator::value_type **const_iterator**
- typedef node_iterator::value_type **iterator**
- typedef base_type::key_const_reference **key_const_reference**
- typedef base_type::key_type **key_type**
- typedef size_type **metadata_type**
- typedef Node_CItr **node_const_iterator**
- typedef Node_Itr **node_iterator**
- typedef allocator_type::size_type **size_type**

Public Member Functions

- iterator find_by_order (size_type)
- const_iterator find_by_order (size_type) const
- size_type order_of_key (key_const_reference) const
- size_type order_of_prefix (a_const_iterator, a_const_iterator) const

Protected Member Functions

- void operator() (node_iterator, node_const_iterator) const

Private Member Functions

- virtual const_iterator **end** () const =0
- virtual const access_traits & **get_access_traits** () const =0

5.957.1 Detailed Description

template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
class __gnu_pbds::trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >

Functor updating ranks of entrees.

5.957.2 Member Function Documentation

find_by_order() [1/2]

```
template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::iterator __gnu_pbds::trie_order_stati
Node_CItr, Node_Itr, _ATraits, _Alloc >::find_by_order (
            size_type order)  [inline]
```

Finds an entry by __order. Returns an iterator to the entry with the __order order, or an iterator to the container object's end if order is at least the size of the container object.

find_by_order() [2/2]

```
template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::const_iterator __gnu_pbds::trie_order
Node_CItr, Node_Itr, _ATraits, _Alloc >::find_by_order (
            size_type order) const  [inline]
```

Finds an entry by __order. Returns a const_iterator to the entry with the __order order, or a const_iterator to the container object's end if order is at least the size of the container object.

operator()()

```
template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
void __gnu_pbds::trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >↩
::operator() (
            node_iterator nd_it,
            node_const_iterator ) const  [inline], [protected]
```

Updates the rank of a node through a node_iterator node_it; end_nd_it is the end node iterator.

order_of_key()

```
template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::size_type __gnu_pbds::trie_order_stat
Node_CItr, Node_Itr, _ATraits, _Alloc >::order_of_key (
            key_const_reference r_key) const  [inline]
```

Returns the order of a key within a sequence. For exapmle, if r_key is the smallest key, this method will return 0; if r_key is a key between the smallest and next key, this method will return 1; if r_key is a key larger than the largest key, this method will return the size of r_c.

order_of_prefix()

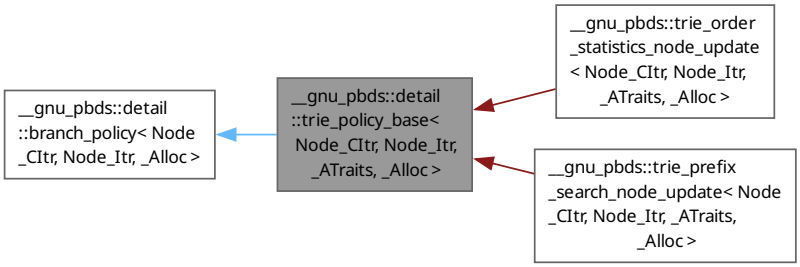
```
template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::size_type __gnu_pbds::trie_order_stat
Node_CItr, Node_Itr, _ATraits, _Alloc >::order_of_prefix (
            a_const_iterator b,
            a_const_iterator e) const  [inline]
```

Returns the order of a prefix within a sequence. For exapmle, if [b, e] is the smallest prefix, this method will return 0; if r_key is a key between the smallest and next key, this method will return 1; if r_key is a key larger than the largest key, this method will return the size of r_c.
The documentation for this class was generated from the following file:

- trie_policy.hpp

5.958 __gnu_pbds::detail::trie_policy_base< Node_CItr, Node_Itr, _ATraits, _Alloc > Class Template Reference

`#include <trie_policy_base.hpp>`
Inheritance diagram for __gnu_pbds::detail::trie_policy_base< Node_CItr, Node_Itr, _ATraits, _Alloc >:



Public Types

- typedef _ATraits **access_traits**
- typedef _Alloc **allocator_type**
- typedef node_const_iterator::value_type **const_iterator**
- typedef node_iterator::value_type **iterator**
- typedef base_type::key_const_reference **key_const_reference**
- typedef base_type::key_type **key_type**
- typedef null_type **metadata_type**
- typedef Node_CItr **node_const_iterator**
- typedef Node_Itr **node_iterator**
- typedef allocator_type::size_type **size_type**

Protected Types

- typedef rebind_v::const_pointer **const_pointer**
- typedef rebind_v::const_reference **const_reference**
- typedef Node_Itr::value_type **it_type**
- typedef remove_const< key_type >::type **rckey_type**
- typedef remove_const< value_type >::type **rcvalue_type**

- typedef rebind_traits< _Alloc, rckey_type > **rebind_k**
- typedef rebind_traits< _Alloc, rcvalue_type > **rebind_v**
- typedef rebind_v::reference **reference**
- typedef std::iterator_traits< it_type >::value_type **value_type**

Protected Member Functions

- virtual const_iterator **end** () const =0
- virtual iterator end ()=0
- it_type **end_iterator** () const
- virtual const access_traits & **get_access_traits** () const =0
- virtual node_const_iterator **node_begin** () const =0
- virtual node_iterator **node_begin** ()=0
- virtual node_const_iterator **node_end** () const =0
- virtual node_iterator **node_end** ()=0

Static Protected Member Functions

- static size_type **common_prefix_len** (node_iterator, e_const_iterator, e_const_iterator, const access_traits &)
- static key_const_reference **extract_key** (const_reference r_val)
- static iterator **leftmost_it** (node_iterator)
- static bool **less** (e_const_iterator, e_const_iterator, e_const_iterator, e_const_iterator, const access_traits &)
- static iterator **rightmost_it** (node_iterator)

5.958.1 Detailed Description

template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
class __gnu_pbds::detail::trie_policy_base< Node_CItr, Node_Itr, _ATraits, _Alloc >

Base class for trie policies.

5.958.2 Member Function Documentation

end()

```
template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
virtual iterator __gnu_pbds::detail::trie_policy_base< Node_CItr, Node_Itr, _ATraits, _Alloc >↩
::end ()  [protected], [pure virtual]
```

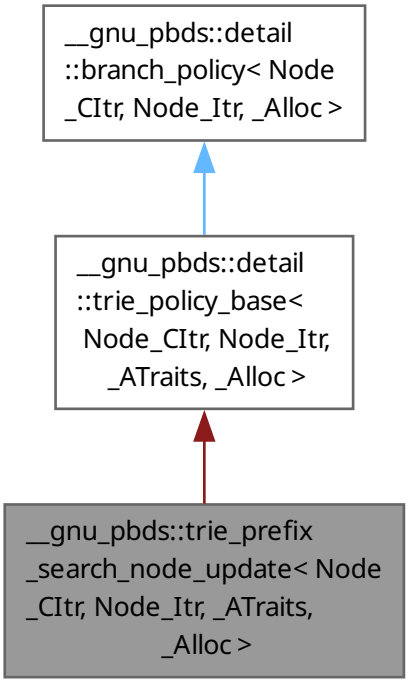
Implements __gnu_pbds::detail::branch_policy< Node_CItr, Node_Itr, _Alloc >.
The documentation for this class was generated from the following file:

- trie_policy_base.hpp

5.959 __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc > Class Template Reference

```
#include <trie_policy.hpp>
```

Inheritance diagram for __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >:



Public Types

- typedef access_traits::const_iterator a_const_iterator
- typedef _ATraits access_traits
- typedef _Alloc allocator_type
- typedef node_const_iterator::value_type **const_iterator**
- typedef node_iterator::value_type **iterator**
- typedef base_type::key_const_reference **key_const_reference**
- typedef base_type::key_type **key_type**
- typedef null_type **metadata_type**
- typedef Node_CItr **node_const_iterator**
- typedef Node_Itr **node_iterator**
- typedef allocator_type::size_type size_type

Public Member Functions

- std::pair< iterator, iterator > prefix_range (a_const_iterator, a_const_iterator)
- std::pair< const_iterator, const_iterator > prefix_range (a_const_iterator, a_const_iterator) const
- std::pair< iterator, iterator > prefix_range (key_const_reference)
- std::pair< const_iterator, const_iterator > prefix_range (key_const_reference) const

Protected Member Functions

- void operator() (node_iterator node_it, node_const_iterator end_nd_it) const

5.959.1 Detailed Description

template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
class __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >

A node updator that allows tries to be searched for the range of values that match a certain prefix.

5.959.2 Member Typedef Documentation

a_const_iterator

```
template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
typedef access_traits::const_iterator __gnu_pbds::trie_prefix_search_node_update< Node_CItr,
Node_Itr, _ATraits, _Alloc >::a_const_iterator
```

Const element iterator.

access_traits

```
template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
typedef _ATraits __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _↩
Alloc >::access_traits
```

Element access traits.

allocator_type

```
template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
typedef _Alloc __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc
>::allocator_type
```

_Alloc type.

size_type

```
template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
typedef allocator_type::size_type __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_↩
Itr, _ATraits, _Alloc >::size_type
```

Size type.

5.959.3 Member Function Documentation

operator()()

```
template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
void __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::operator()
(
            node_iterator node_it,
            node_const_iterator end_nd_it) const  [inline], [protected]
```

Called to update a node's metadata.

prefix_range() [1/4]

```
template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
std::pair< typename trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >↩
::iterator, typename trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >↩
```

```
::iterator > __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc
>::prefix_range (
              a_const_iterator b,
              a_const_iterator e)
```

Finds the iterator range corresponding to all values whose prefixes match [b, e).

prefix_range() [2/4]

```
template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
std::pair< typename trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >↩
::const_iterator, typename trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc
>::const_iterator > __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, ↩
_Alloc >::prefix_range (
              a_const_iterator b,
              a_const_iterator e) const
```

Finds the const iterator range corresponding to all values whose prefixes match [b, e).

prefix_range() [3/4]

```
template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
std::pair< typename trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >↩
::iterator, typename trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >↩
::iterator > __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc
>::prefix_range (
              key_const_reference r_key)
```

Finds the iterator range corresponding to all values whose prefixes match r_key.

prefix_range() [4/4]

```
template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
std::pair< typename trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >↩
::const_iterator, typename trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc
>::const_iterator > __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, ↩
_Alloc >::prefix_range (
              key_const_reference r_key) const
```

Finds the const iterator range corresponding to all values whose prefixes match r_key.
The documentation for this class was generated from the following file:

- trie_policy.hpp

5.960 __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc > Struct Template Reference

```
#include <trie_policy.hpp>
```

Public Types

- enum { **reverse** }
- enum { **min_e_val** , **max_e_val** , **max_size** }
- typedef detail::__conditional_type< Reverse, typenameString::const_reverse_iterator, typenameString::const_↩ iterator >::__type const_iterator
- typedef std::iterator_traits< const_iterator >::value_type e_type
- typedef detail::rebind_traits< _Alloc, key_type >::const_reference **key_const_reference**
- typedef String **key_type**
- typedef _Alloc::size_type **size_type**

Static Public Member Functions

- static const_iterator begin (key_const_reference)
- static size_type e_pos (e_type e)
- static const_iterator end (key_const_reference)

5.960.1 Detailed Description

template<typename String = std::string, typename String::value_type Min_E_Val = detail::__numeric_traits<typename String::value_type>::__min, typename String::value_type Max_E_Val = detail::__numeric_traits<typename String::value_type>::__max, bool Reverse = false, typename _Alloc = std::allocator<char>> struct __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >

Element access traits for string types.

Template Parameters

String	String type.
Min_E_Val	Minimal element value.
Max_E_Val	Maximum element value.
Reverse	Reverse iteration should be used. Default: false.
_Alloc	Allocator type.

5.960.2 Member Typedef Documentation

const_iterator

```
template<typename String = std::string, typename String::value_type Min_E_Val = detail::__numeric_traits<typename String::value_type>::__min, typename String::value_type Max_E_Val = detail::__numeric_traits<typename String::value_type>::__max, bool Reverse = false, typename _Alloc = std::allocator<char>>
typedef detail::__conditional_type<Reverse,typenameString::const_reverse_iterator,typenameString::const_iterator>::__type __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::const_iterator
```

Element const iterator type.

e_type

```
template<typename String = std::string, typename String::value_type Min_E_Val = detail::__numeric_traits<typename String::value_type>::__min, typename String::value_type Max_E_Val = detail::__numeric_traits<typename String::value_type>::__max, bool Reverse = false, typename _Alloc = std::allocator<char>>
typedef std::iterator_traits<const_iterator>::value_type __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::e_type
```

Element type.

5.960.3 Member Function Documentation

begin()

```
template<typename String, typename String::value_type Min_E_Val, typename String::value_type Max_E_Val, bool Reverse, typename _Alloc>
trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::const_iterator __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::begin (
            key_const_reference r_key)  [inline], [static]
```

Returns a const_iterator to the first element of key_const_reference agumnet.

e_pos()

```
template<typename String, typename String::value_type Min_E_Val, typename String::value_type Max↩
_E_Val, bool Reverse, typename _Alloc>
trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::size_type __gnu_pbds::trie_string_acc
String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::e_pos (
            e_type e)  [inline], [static]
```

Maps an element to a position.

end()

```
template<typename String, typename String::value_type Min_E_Val, typename String::value_type Max↩
_E_Val, bool Reverse, typename _Alloc>
trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::const_iterator __gnu_pbds::trie_strin
String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::end (
            key_const_reference r_key)  [inline], [static]
```

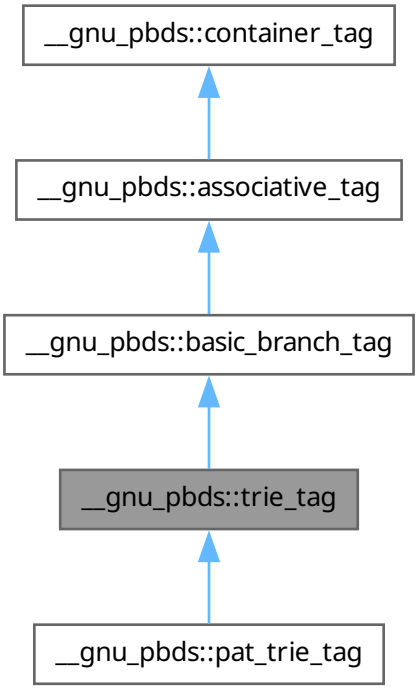
Returns a const_iterator to the after-last element of key_const_reference argument.
The documentation for this struct was generated from the following file:

- trie_policy.hpp

5.961 __gnu_pbds::trie_tag Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for __gnu_pbds::trie_tag:



5.961.1 Detailed Description

Basic trie structure.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

5.962 __gnu_pbds::detail::trie_traits< Key, Data, _ATraits, Node_Update, Tag, _Alloc > Struct Template Reference

5.962.1 Detailed Description

template<typename Key, typename Data, typename _ATraits, template< typename Node_CItr, typename Node_Itr, typename _ATraits_, typename _Alloc > class Node_Update, typename Tag, typename _Alloc> struct __gnu_pbds::detail::trie_traits< Key, Data, _ATraits, Node_Update, Tag, _Alloc >

Trie traits class, primary template.
The documentation for this struct was generated from the following file:

- branch_policy/traits.hpp

5.963 __gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc > Struct Template Reference

`#include <traits.hpp>`

Public Types

- typedef _ATraits **access_traits**
- typedef base_type::_CIter< node, leaf, head, inode, true > **const_iterator**
- typedef base_type::_CIter< node, leaf, head, inode, false > **const_reverse_iterator**
- typedef base_type::_Head< synth_access_traits, metadata > **head**
- typedef base_type::_Inode< synth_access_traits, metadata > **inode**
- typedef base_type::_Iter< node, leaf, head, inode, true > **iterator**
- typedef base_type::_Leaf< synth_access_traits, metadata > **leaf**
- typedef base_type::_Metadata< metadata_type, _Alloc > **metadata**
- typedef trie_node_metadata_dispatch< Key, Mapped, _ATraits, Node_Update, _Alloc >::type **metadata_type**
- typedef base_type::_Node_base< synth_access_traits, metadata > **node**
- typedef base_type::_Node_citer< node, leaf, head, inode, const_iterator, iterator, _Alloc > node_const_iterator
- typedef base_type::_Node_iter< node, leaf, head, inode, const_iterator, iterator, _Alloc > **node_iterator**
- typedef Node_Update< node_const_iterator, node_iterator, _ATraits, _Alloc > node_update
- typedef null_node_update< node_const_iterator, node_iterator, _ATraits, _Alloc > * **null_node_update_pointer**
- typedef base_type::_Iter< node, leaf, head, inode, false > **reverse_iterator**
- typedef __gnu_pbds::detail::synth_access_traits< type_traits, false, access_traits > synth_access_traits

5.963.1 Detailed Description

template<typename Key, typename Mapped, typename _ATraits, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >

Specialization.

5.963.2 Member Typedef Documentation

node_const_iterator

```
template<typename Key, typename Mapped, typename _ATraits, template< typename Node_CItr, typename
Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
typedef base_type::_Node_citer<node, leaf, head, inode, const_iterator, iterator, _Alloc> __gnu_pbds::detail::tri
Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >::node_const_iterator
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

node_update

```
template<typename Key, typename Mapped, typename _ATraits, template< typename Node_CItr, typename
Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
typedef Node_Update<node_const_iterator, node_iterator, _ATraits, _Alloc> __gnu_pbds::detail::trie_traits<
Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >::node_update
```

Type for node update.

synth_access_traits

```
template<typename Key, typename Mapped, typename _ATraits, template< typename Node_CItr, typename
Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
typedef __gnu_pbds::detail::synth_access_traits<type_traits, false, access_traits> __gnu_pbds::detail::trie_trait
Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >::synth_access_traits
```

Type for synthesized traits.

The documentation for this struct was generated from the following file:

- pat_trie_/traits.hpp

5.964 __gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc > Struct Template Reference

```
#include <traits.hpp>
```

Public Types

- typedef _ATraits **access_traits**
- typedef base_type::_CIter< node, leaf, head, inode, true > **const_iterator**
- typedef base_type::_CIter< node, leaf, head, inode, false > **const_reverse_iterator**
- typedef base_type::_Head< synth_access_traits, metadata > **head**
- typedef base_type::_Inode< synth_access_traits, metadata > **inode**
- typedef const_iterator **iterator**
- typedef base_type::_Leaf< synth_access_traits, metadata > **leaf**
- typedef base_type::_Metadata< metadata_type, _Alloc > **metadata**
- typedef trie_node_metadata_dispatch< Key, null_type, _ATraits, Node_Update, _Alloc >::type **metadata_type**
- typedef base_type::_Node_base< synth_access_traits, metadata > **node**
- typedef base_type::_Node_citer< node, leaf, head, inode, const_iterator, iterator, _Alloc > node_const_iterator
- typedef node_const_iterator **node_iterator**
- typedef Node_Update< node_const_iterator, node_iterator, _ATraits, _Alloc > node_update
- typedef null_node_update< node_const_iterator, node_const_iterator, _ATraits, _Alloc > * **null_node_update_pointer**
- typedef const_reverse_iterator **reverse_iterator**
- typedef __gnu_pbds::detail::synth_access_traits< type_traits, true, access_traits > synth_access_traits

5.964.1 Detailed Description

template<typename Key, typename _ATraits, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >

Specialization.

5.964.2 Member Typedef Documentation

node_const_iterator

```
template<typename Key, typename _ATraits, template< typename Node_CItr, typename Node_Itr, typename
Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
typedef base_type::_Node_citer<node, leaf, head, inode, const_iterator, iterator, _Alloc> __gnu_pbds::detail::tri
Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >::node_const_iterator
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

node_update

```
template<typename Key, typename _ATraits, template< typename Node_CItr, typename Node_Itr, typename
Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
typedef Node_Update<node_const_iterator, node_iterator, _ATraits, _Alloc> __gnu_pbds::detail::trie_traits<
Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >::node_update
```

Type for node update.

synth_access_traits

```
template<typename Key, typename _ATraits, template< typename Node_CItr, typename Node_Itr, typename
Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
typedef __gnu_pbds::detail::synth_access_traits<type_traits, true, access_traits> __gnu_pbds::detail::trie_traits
Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >::synth_access_traits
```

Type for synthesized traits.
The documentation for this struct was generated from the following file:

- pat_trie_/traits.hpp

5.965 __gnu_pbds::trivial_iterator_tag Struct Reference

```
#include <tag_and_trait.hpp>
```

5.965.1 Detailed Description

A trivial iterator tag. Signifies that the iterators has none of std::iterators's movement abilities.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

5.966 std::try_to_lock_t Struct Reference

```
#include <std_mutex.h>
```

5.966.1 Detailed Description

Try to acquire ownership of the mutex without blocking.
The documentation for this struct was generated from the following file:

- std_mutex.h

5.967 std::tuple< _Elements > Class Template Reference

```
#include <tuple>
```

Public Member Functions

- template<typename... _UElements, bool _Valid = __valid_args<_UElements...>(), _ImplicitCtor< _Valid, _UElements... > = true>
  constexpr **tuple** (_UElements &&... __elements) noexcept(__nothrow_constructible< _UElements... >())
- template<typename... _UElements, bool _Valid = __valid_args<_UElements...>(), _ExplicitCtor< _Valid, _UElements... > = false>
  constexpr **tuple** (_UElements &&... __elements) noexcept(__nothrow_constructible< _UElements... >())
- template<typename _Alloc, _ImplicitDefaultCtor< is_object< _Alloc >::value > = true>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a)
- template<typename _Alloc, _ExplicitDefaultCtor< is_object< _Alloc >::value > = false>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a)

- template<typename _Alloc, typename... _UElements, bool _Valid = __valid_args<_UElements...>(), _ImplicitCtor< _Valid, _UElements... > = true>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, _UElements &&... __elements)
- template<typename _Alloc, typename... _UElements, bool _Valid = __valid_args<_UElements...>(), _ExplicitCtor< _Valid, _UElements... > = false>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, _UElements &&... __elements)
- template<typename _Alloc, bool _NotEmpty = (sizeof...(_Elements) >= 1), _ImplicitCtor< _NotEmpty, const _Elements &... > = true>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, const _Elements &... __elements)
- template<typename _Alloc, bool _NotEmpty = (sizeof...(_Elements) >= 1), _ExplicitCtor< _NotEmpty, const _Elements &... > = false>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, const _Elements &... __elements)
- template<typename _Alloc>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, const tuple &__in)
- template<typename _Alloc, typename... _UElements, bool _Valid = (sizeof...(_Elements) == sizeof...(_UElements)) && !__use_other_ctor<const tuple<_UElements...>&>(), _ImplicitCtor< _Valid, const _UElements &... > = true>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, const tuple< _UElements... > &__in)
- template<typename _Alloc, typename... _UElements, bool _Valid = (sizeof...(_Elements) == sizeof...(_UElements)) && !__use_other_ctor<const tuple<_UElements...>&>(), _ExplicitCtor< _Valid, const _UElements &... > = false>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, const tuple< _UElements... > &__in)
- template<typename _Alloc>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, tuple &&__in)
- template<typename _Alloc, typename... _UElements, bool _Valid = (sizeof...(_Elements) == sizeof...(_UElements)) && !__use_other_ctor<tuple<_UElements...>&&>(), _ImplicitCtor< _Valid, _UElements... > = true>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, tuple< _UElements... > &&__in)
- template<typename _Alloc, typename... _UElements, bool _Valid = (sizeof...(_Elements) == sizeof...(_UElements)) && !__use_other_ctor<tuple<_UElements...>&&>(), _ExplicitCtor< _Valid, _UElements... > = false>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, tuple< _UElements... > &&__in)
- template<bool _NotEmpty = (sizeof...(_Elements) >= 1), _ImplicitCtor< _NotEmpty, const _Elements &... > = true>
  constexpr **tuple** (const _Elements &... __elements) noexcept(__nothrow_constructible< const _Elements &... >())
- template<bool _NotEmpty = (sizeof...(_Elements) >= 1), _ExplicitCtor< _NotEmpty, const _Elements &... > = false>
  constexpr **tuple** (const _Elements &... __elements) noexcept(__nothrow_constructible< const _Elements &... >())
- constexpr **tuple** (const tuple &)=default
- template<typename... _UElements, bool _Valid = (sizeof...(_Elements) == sizeof...(_UElements)) && !__use_other_ctor<const tuple<_UElements...>&>(), _ImplicitCtor< _Valid, const _UElements &... > = true>
  constexpr **tuple** (const tuple< _UElements... > &__in) noexcept(__nothrow_constructible< const _UElements &... >())
- template<typename... _UElements, bool _Valid = (sizeof...(_Elements) == sizeof...(_UElements)) && !__use_other_ctor<const tuple<_UElements...>&>(), _ExplicitCtor< _Valid, const _UElements &... > = false>
  constexpr **tuple** (const tuple< _UElements... > &__in) noexcept(__nothrow_constructible< const _UElements &... >())
- constexpr **tuple** (tuple &&)=default
- template<typename... _UElements, bool _Valid = (sizeof...(_Elements) == sizeof...(_UElements)) && !__use_other_ctor<tuple<_UElements...>&&>(), _ImplicitCtor< _Valid, _UElements... > = true>
  constexpr **tuple** (tuple< _UElements... > &&__in) noexcept(__nothrow_constructible< _UElements... >())
- template<typename... _UElements, bool _Valid = (sizeof...(_Elements) == sizeof...(_UElements)) && !__use_other_ctor<tuple<_UElements...>&&>(), _ExplicitCtor< _Valid, _UElements... > = false>
  constexpr **tuple** (tuple< _UElements... > &&__in) noexcept(__nothrow_constructible< _UElements... >())
- constexpr tuple & **operator=** (__conditional_t< __assignable< _Elements... >(), tuple &&, __nonesuch && > __in) noexcept(__nothrow_assignable< _Elements... >())
- constexpr tuple & **operator=** (__conditional_t< __assignable< const _Elements &... >(), const tuple &, const __nonesuch & > __in) noexcept(__nothrow_assignable< const _Elements &... >())

- template<typename... _UElements>
  constexpr __enable_if_t< __assignable< const _UElements &... >(), tuple & > **operator=** (const tuple< _UElements... > &__in) noexcept(__nothrow_assignable< const _UElements &... >())
- template<typename... _UElements>
  constexpr __enable_if_t< __assignable< _UElements... >(), tuple & > **operator=** (tuple< _UElements... > &&__in) noexcept(__nothrow_assignable< _UElements... >())
- constexpr void **swap** (tuple &__in) noexcept(__and_< __is_nothrow_swappable< _Elements >... >::value)

5.967.1 Detailed Description

template<typename... _Elements>
class std::tuple< _Elements >

Primary class template, tuple.
The documentation for this class was generated from the following file:

- tuple

5.968 std::tuple< _T1, _T2 > Class Template Reference

`#include <tuple>`

Public Member Functions

- template<typename _U1, typename _U2, _ImplicitCtor<!__is_alloc_arg< _U1 >(), _U1, _U2 > = true>
  constexpr **tuple** (_U1 &&__a1, _U2 &&__a2) noexcept(__nothrow_constructible< _U1, _U2 >())
- template<typename _U1, typename _U2, _ExplicitCtor<!__is_alloc_arg< _U1 >(), _U1, _U2 > = false>
  constexpr **tuple** (_U1 &&__a1, _U2 &&__a2) noexcept(__nothrow_constructible< _U1, _U2 >())
- constexpr **tuple** (_UElements &&... __elements) noexcept(__nothrow_constructible< _UElements... >())
- constexpr **tuple** (_UElements &&... __elements) noexcept(__nothrow_constructible< _UElements... >())
- constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a)
- constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a)
- template<typename _Alloc, _ImplicitDefaultCtor< is_object< _Alloc >::value, _T1, _T2 > = true>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a)
- template<typename _Alloc, _ExplicitDefaultCtor< is_object< _Alloc >::value, _T1, _T2 > = false>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a)
- template<typename _Alloc, typename _U1, typename _U2, _ImplicitCtor< true, _U1, _U2 > = true>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, _U1 &&__a1, _U2 &&__a2)
- template<typename _Alloc, typename _U1, typename _U2, _ExplicitCtor< true, _U1, _U2 > = false>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, _U1 &&__a1, _U2 &&__a2)
- constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, _UElements &&... __elements)
- constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, _UElements &&... __elements)
- constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, const _Elements &... __elements)
- constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, const _Elements &... __elements)
- template<typename _Alloc, bool _Dummy = true, _ImplicitCtor< _Dummy, const _T1 &, const _T2 & > = true>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, const _T1 &__a1, const _T2 &__a2)
- template<typename _Alloc, bool _Dummy = true, _ExplicitCtor< _Dummy, const _T1 &, const _T2 & > = false>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, const _T1 &__a1, const _T2 &__a2)
- template<typename _Alloc, typename _U1, typename _U2, _ImplicitCtor< true, const _U1 &, const _U2 & > = true>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, const pair< _U1, _U2 > &__in)
- template<typename _Alloc, typename _U1, typename _U2, _ExplicitCtor< true, const _U1 &, const _U2 & > = false>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, const pair< _U1, _U2 > &__in)
- constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, const tuple &__in)

- template<typename _Alloc>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, const tuple &__in)
- template<typename _Alloc, typename _U1, typename _U2, _ImplicitCtor< true, const _U1 &, const _U2 & > = true>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, const tuple< _U1, _U2 > &__in)
- template<typename _Alloc, typename _U1, typename _U2, _ExplicitCtor< true, const _U1 &, const _U2 & > = false>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, const tuple< _U1, _U2 > &__in)
- constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, const tuple< _UElements... > &__in)
- constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, const tuple< _UElements... > &__in)
- template<typename _Alloc, typename _U1, typename _U2, _ImplicitCtor< true, _U1, _U2 > = true>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, pair< _U1, _U2 > &&__in)
- template<typename _Alloc, typename _U1, typename _U2, _ExplicitCtor< true, _U1, _U2 > = false>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, pair< _U1, _U2 > &&__in)
- constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, tuple &&__in)
- template<typename _Alloc>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, tuple &&__in)
- template<typename _Alloc, typename _U1, typename _U2, _ImplicitCtor< true, _U1, _U2 > = true>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, tuple< _U1, _U2 > &&__in)
- template<typename _Alloc, typename _U1, typename _U2, _ExplicitCtor< true, _U1, _U2 > = false>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, tuple< _U1, _U2 > &&__in)
- constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, tuple< _UElements... > &&__in)
- constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, tuple< _UElements... > &&__in)
- constexpr **tuple** (const _Elements &... __elements) noexcept(__nothrow_constructible< const _Elements &... >())
- constexpr **tuple** (const _Elements &... __elements) noexcept(__nothrow_constructible< const _Elements &... >())
- template<bool _Dummy = true, _ImplicitCtor< _Dummy, const _T1 &, const _T2 & > = true>
  constexpr **tuple** (const _T1 &__a1, const _T2 &__a2) noexcept(__nothrow_constructible< const _T1 &, const _T2 & >())
- template<bool _Dummy = true, _ExplicitCtor< _Dummy, const _T1 &, const _T2 & > = false>
  constexpr **tuple** (const _T1 &__a1, const _T2 &__a2) noexcept(__nothrow_constructible< const _T1 &, const _T2 & >())
- template<typename _U1, typename _U2, _ImplicitCtor< true, const _U1 &, const _U2 & > = true>
  constexpr **tuple** (const pair< _U1, _U2 > &__in) noexcept(__nothrow_constructible< const _U1 &, const _U2 & >())
- template<typename _U1, typename _U2, _ExplicitCtor< true, const _U1 &, const _U2 & > = false>
  constexpr **tuple** (const pair< _U1, _U2 > &__in) noexcept(__nothrow_constructible< const _U1 &, const _U2 & >())
- constexpr **tuple** (const tuple &)=default
- constexpr **tuple** (const tuple &)=default
- template<typename _U1, typename _U2, _ImplicitCtor< true, const _U1 &, const _U2 & > = true>
  constexpr **tuple** (const tuple< _U1, _U2 > &__in) noexcept(__nothrow_constructible< const _U1 &, const _U2 & >())
- template<typename _U1, typename _U2, _ExplicitCtor< true, const _U1 &, const _U2 & > = false>
  constexpr **tuple** (const tuple< _U1, _U2 > &__in) noexcept(__nothrow_constructible< const _U1 &, const _U2 & >())
- constexpr **tuple** (const tuple< _UElements... > &__in) noexcept(__nothrow_constructible< const _UElements &... >())
- constexpr **tuple** (const tuple< _UElements... > &__in) noexcept(__nothrow_constructible< const _UElements &... >())
- template<typename _U1, typename _U2, _ImplicitCtor< true, _U1, _U2 > = true>
  constexpr **tuple** (pair< _U1, _U2 > &&__in) noexcept(__nothrow_constructible< _U1, _U2 >())
- template<typename _U1, typename _U2, _ExplicitCtor< true, _U1, _U2 > = false>
  constexpr **tuple** (pair< _U1, _U2 > &&__in) noexcept(__nothrow_constructible< _U1, _U2 >())

- constexpr **tuple** (tuple &&)=default
- constexpr **tuple** (tuple &&)=default
- template<typename _U1, typename _U2, _ImplicitCtor< true, _U1, _U2 > = true>
  constexpr **tuple** (tuple< _U1, _U2 > &&__in) noexcept(__nothrow_constructible< _U1, _U2 >())
- template<typename _U1, typename _U2, _ExplicitCtor< true, _U1, _U2 > = false>
  constexpr **tuple** (tuple< _U1, _U2 > &&__in) noexcept(__nothrow_constructible< _U1, _U2 >())
- constexpr **tuple** (tuple< _UElements... > &&__in) noexcept(__nothrow_constructible< _UElements... >())
- constexpr **tuple** (tuple< _UElements... > &&__in) noexcept(__nothrow_constructible< _UElements... >())
- constexpr tuple & **operator=** (__conditional_t< __assignable< _Elements... >(), tuple &&, __nonesuch && > __in) noexcept(__nothrow_assignable< _Elements... >())
- constexpr tuple & **operator=** (__conditional_t< __assignable< _T1, _T2 >(), tuple &&, __nonesuch && > __in) noexcept(__nothrow_assignable< _T1, _T2 >())
- constexpr tuple & **operator=** (__conditional_t< __assignable< const _Elements &... >(), const tuple &, const __nonesuch & > __in) noexcept(__nothrow_assignable< const _Elements &... >())
- constexpr tuple & **operator=** (__conditional_t< __assignable< const _T1 &, const _T2 & >(), const tuple &, const __nonesuch & > __in) noexcept(__nothrow_assignable< const _T1 &, const _T2 & >())
- template<typename _U1, typename _U2>
  constexpr __enable_if_t< __assignable< const _U1 &, const _U2 & >(), tuple & > **operator=** (const pair< _U1, _U2 > &__in) noexcept(__nothrow_assignable< const _U1 &, const _U2 & >())
- template<typename _U1, typename _U2>
  constexpr __enable_if_t< __assignable< const _U1 &, const _U2 & >(), tuple & > **operator=** (const tuple< _U1, _U2 > &__in) noexcept(__nothrow_assignable< const _U1 &, const _U2 & >())
- constexpr __enable_if_t< __assignable< const _UElements &... >(), tuple & > **operator=** (const tuple< _↩ UElements... > &__in) noexcept(__nothrow_assignable< const _UElements &... >())
- template<typename _U1, typename _U2>
  constexpr __enable_if_t< __assignable< _U1, _U2 >(), tuple & > **operator=** (pair< _U1, _U2 > &&__in) noexcept(__nothrow_assignable< _U1, _U2 >())
- template<typename _U1, typename _U2>
  constexpr __enable_if_t< __assignable< _U1, _U2 >(), tuple & > **operator=** (tuple< _U1, _U2 > &&__in) noexcept(__nothrow_assignable< _U1, _U2 >())
- constexpr __enable_if_t< __assignable< _UElements... >(), tuple & > **operator=** (tuple< _UElements... > &&__in) noexcept(__nothrow_assignable< _UElements... >())
- constexpr void **swap** (tuple &__in) noexcept(__and_< __is_nothrow_swappable< _Elements >... >::value)
- constexpr void **swap** (tuple &__in) noexcept(__and_< __is_nothrow_swappable< _T1 >, __is_nothrow_↩ swappable< _T2 > >::value)

5.968.1 Detailed Description

template<typename _T1, typename _T2>
class std::tuple< _T1, _T2 >

Partial specialization, 2-element tuple. Includes construction and assignment from a pair.
The documentation for this class was generated from the following file:

- tuple

5.969 std::tuple_element< __i, _Tp > Struct Template Reference

5.969.1 Detailed Description

template<size_t __i, typename _Tp>
struct std::tuple_element< __i, _Tp >

Gives the type of the ith element of a given tuple type.
The documentation for this struct was generated from the following file:

- utility.h

5.970 std::tuple_element< 0, pair< _Tp1, _Tp2 > > Struct Template Reference

`#include <stl_pair.h>`

Public Types

- typedef _Tp1 **type**

5.970.1 Detailed Description

template<class _Tp1, class _Tp2>
struct std::tuple_element< 0, pair< _Tp1, _Tp2 > >

Partial specialization for std::pair.
The documentation for this struct was generated from the following file:

- stl_pair.h

5.971 std::tuple_element< 1, pair< _Tp1, _Tp2 > > Struct Template Reference

`#include <stl_pair.h>`

Public Types

- typedef _Tp2 **type**

5.971.1 Detailed Description

template<class _Tp1, class _Tp2>
struct std::tuple_element< 1, pair< _Tp1, _Tp2 > >

Partial specialization for std::pair.
The documentation for this struct was generated from the following file:

- stl_pair.h

5.972 std::tuple_element< __i, tuple< _Types... > > Struct Template Reference

`#include <tuple>`

Public Types

- using **type**

5.972.1 Detailed Description

template<size_t __i, typename... _Types>
struct std::tuple_element< __i, tuple< _Types... > >

Trait to get the Ith element type from a tuple.
The documentation for this struct was generated from the following file:

- tuple

5.973 std::tuple_element< _Ind, array< _Tp, _Nm > > Struct Template Reference

`#include <array>`

Public Types

- using **type**

5.973.1 Detailed Description

template<size_t _Ind, typename _Tp, size_t _Nm>
struct std::tuple_element< _Ind, array< _Tp, _Nm > >

Partial specialization for std::array.
The documentation for this struct was generated from the following file:

- array

5.974 std::tuple_size< _Tp > Struct Template Reference

Inherited by std::tuple_size< const __enable_if_has_tuple_size< _Tp > >, std::tuple_size< const volatile __enable↩ _if_has_tuple_size< _Tp > >, and std::tuple_size< volatile __enable_if_has_tuple_size< _Tp > >.

5.974.1 Detailed Description

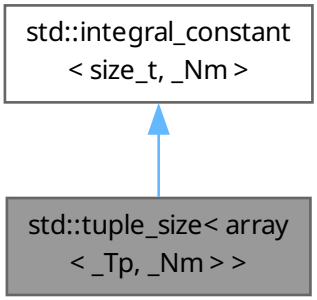
template<typename _Tp>
struct std::tuple_size< _Tp >

Finds the size of a given tuple type.
The documentation for this struct was generated from the following file:

- utility.h

5.975 std::tuple_size< array< _Tp, _Nm > > Struct Template Reference

`#include <array>`
Inheritance diagram for std::tuple_size< array< _Tp, _Nm > >:



Public Types

- using **type**
- using **value_type**

Public Member Functions

- constexpr **operator value_type** () const noexcept

Static Public Attributes

- static constexpr size_t **value**

5.975.1 Detailed Description

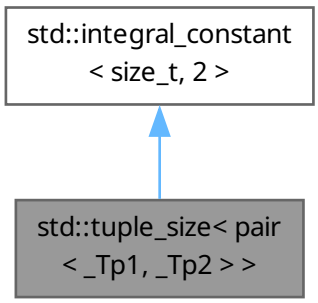
template<**typename _Tp, size_t _Nm**>
struct std::tuple_size< **array**< **_Tp, _Nm** > >

Partial specialization for std::array.
The documentation for this struct was generated from the following file:

- array

5.976 std::tuple_size< pair< _Tp1, _Tp2 > > Struct Template Reference

`#include <stl_pair.h>`
Inheritance diagram for std::tuple_size< pair< _Tp1, _Tp2 > >:



Public Types

- using **type**
- using **value_type**

Public Member Functions

- constexpr **operator value_type** () const noexcept

Static Public Attributes

- static constexpr size_t **value**

5.976.1 Detailed Description

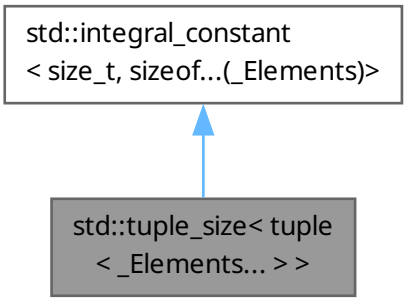
template<class _Tp1, class _Tp2>
struct std::tuple_size< pair< _Tp1, _Tp2 > >

Partial specialization for std::pair.
The documentation for this struct was generated from the following file:

- stl_pair.h

5.977 std::tuple_size< tuple< _Elements... > > Struct Template Reference

`#include <tuple>`
Inheritance diagram for std::tuple_size< tuple< _Elements... > >:



Public Types

- using **type**
- using **value_type**

Public Member Functions

- constexpr **operator value_type** () const noexcept

Static Public Attributes

- static constexpr size_t **value**

5.977.1 Detailed Description

template<typename... _Elements>
struct std::tuple_size< tuple< _Elements... > >

class tuple_size
The documentation for this struct was generated from the following file:

- tuple

5.978 __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, false >::type Struct Reference

`#include <entry_cmp.hpp>`

Public Member Functions

- **type** (const Cmp_Fn &other)
- bool **operator()** (entry lhs, entry rhs) const

5.978.1 Detailed Description

template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, false >::type

Compare plus entry.
The documentation for this struct was generated from the following file:

- entry_cmp.hpp

5.979 std::type_index Struct Reference

`#include <typeindex>`

Public Member Functions

- **type_index** (const type_info &__rhs) noexcept
- size_t **hash_code** () const noexcept
- const char * **name** () const noexcept
- bool **operator!=** (const type_index &__rhs) const noexcept
- bool **operator<** (const type_index &__rhs) const noexcept
- bool **operator<=** (const type_index &__rhs) const noexcept
- bool **operator==** (const type_index &__rhs) const noexcept
- bool **operator>** (const type_index &__rhs) const noexcept
- bool **operator>=** (const type_index &__rhs) const noexcept

5.979.1 Detailed Description

Class type_index.
The class type_index provides a simple wrapper for type_info which can be used as an index type in associative containers (23.6) and in unordered associative containers (23.7).
The documentation for this struct was generated from the following file:

- typeindex

5.980 __gnu_debug::type_info Class Reference

`#include <typeinfo>`

Public Member Functions

- virtual ~type_info ()
- virtual bool **__do_catch** (const type_info *__thr_type, void **__thr_obj, unsigned __outer) const
- virtual bool **__do_upcast** (const __cxxabiv1::__class_type_info *__target, void **__obj_ptr) const
- virtual bool **__is_function_p** () const

- virtual bool **__is_pointer_p** () const
- bool before (const type_info &__arg) const noexcept
- size_t **hash_code** () const noexcept
- const char * name () const noexcept
- constexpr bool **operator==** (const type_info &__arg) const noexcept

Protected Member Functions

- **type_info** (const char *__n)

Protected Attributes

- const char * **__name**

5.980.1 Detailed Description

Part of RTTI.
The `type_info` class describes type information generated by an implementation.

5.980.2 Constructor & Destructor Documentation

~type_info()

```
virtual std::type_info::~type_info ()  [virtual]
```

Destructor first. Being the first non-inline virtual function, this controls in which translation unit the vtable is emitted. The compiler makes use of that information to know where to emit the runtime-mandated type_info structures in the new-abi.

5.980.3 Member Function Documentation

before()

```
bool std::type_info::before (
            const type_info & __arg) const  [noexcept]
```

Returns true if `*this` precedes `__arg` in the implementation's collation order.

name()

```
const char * std::type_info::name () const  [inline], [noexcept]
```

Returns an *implementation-defined* byte string; this is not portable between compilers!
The documentation for this class was generated from the following file:

- typeinfo

5.981 std::type_info Class Reference

`#include <typeinfo>`

Public Member Functions

- virtual ~type_info ()
- virtual bool **__do_catch** (const type_info *__thr_type, void **__thr_obj, unsigned __outer) const
- virtual bool **__do_upcast** (const __cxxabiv1::__class_type_info *__target, void **__obj_ptr) const
- virtual bool **__is_function_p** () const
- virtual bool **__is_pointer_p** () const
- bool before (const type_info &__arg) const noexcept
- size_t **hash_code** () const noexcept

- const char * name () const noexcept
- constexpr bool **operator==** (const type_info &__arg) const noexcept

Protected Member Functions

- **type_info** (const char *__n)

Protected Attributes

- const char * **__name**

5.981.1 Detailed Description

Part of RTTI.
The `type_info` class describes type information generated by an implementation.

5.981.2 Constructor & Destructor Documentation

~type_info()

```
virtual std::type_info::~type_info ()  [virtual]
```

Destructor first. Being the first non-inline virtual function, this controls in which translation unit the vtable is emitted. The compiler makes use of that information to know where to emit the runtime-mandated type_info structures in the new-abi.

5.981.3 Member Function Documentation

before()

```
bool std::type_info::before (
            const type_info & __arg) const  [noexcept]
```

Returns true if `*this` precedes `__arg` in the implementation's collation order.

name()

```
const char * std::type_info::name () const  [inline], [noexcept]
```

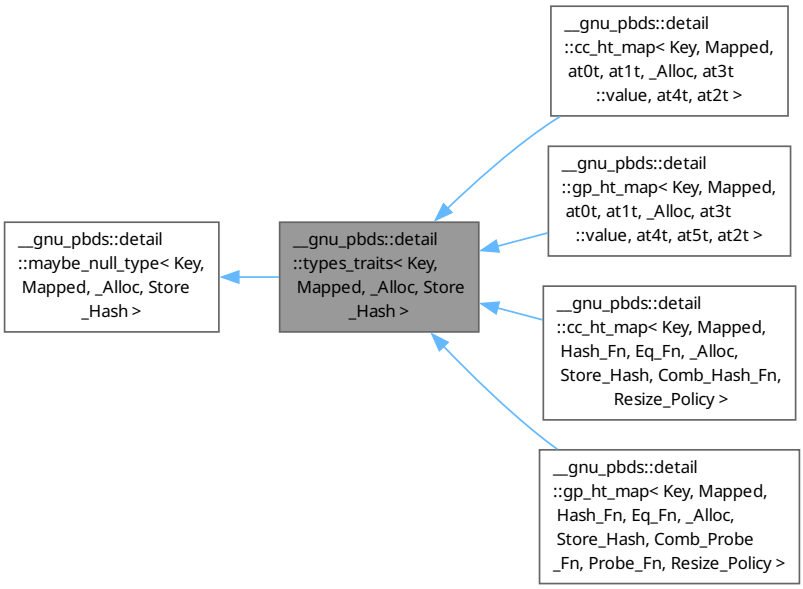
Returns an *implementation-defined* byte string; this is not portable between compilers!
The documentation for this class was generated from the following file:

- typeinfo

5.982 __gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash > Struct Template Reference

```
#include <types_traits.hpp>
```

Inheritance diagram for __gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash >:



Public Types

- typedef std::pair< size_type, size_type > **comp_hash**
- typedef __rebind_va::const_pointer **const_pointer**
- typedef __rebind_va::const_reference **const_reference**
- typedef __rebind_ka::const_pointer **key_const_pointer**
- typedef __rebind_ka::const_reference **key_const_reference**
- typedef __rebind_ka::pointer **key_pointer**
- typedef __rebind_ka::reference **key_reference**
- typedef Key **key_type**
- typedef __rebind_ma::const_pointer **mapped_const_pointer**
- typedef __rebind_ma::const_reference **mapped_const_reference**
- typedef __rebind_ma::pointer **mapped_pointer**
- typedef __rebind_ma::reference **mapped_reference**
- typedef Mapped **mapped_type**
- typedef __nothrowcopy::indicator **no_throw_indicator**
- typedef __rebind_va::pointer **pointer**
- typedef __rebind_va::reference **reference**
- typedef _Alloc::size_type **size_type**
- typedef integral_constant< int, Store_Hash > **store_extra**
- typedef stored_data< value_type, size_type, Store_Hash > **stored_data_type**
- typedef select_value_type< Key, Mapped >::type **value_type**

Public Attributes

- no_throw_indicator **m_no_throw_copies_indicator**
- store_extra **m_store_extra_indicator**

5.982.1 Detailed Description

template<typename Key, typename Mapped, typename _Alloc, bool Store_Hash>
struct __gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash >

Traits for abstract types.
The documentation for this struct was generated from the following file:

- types_traits.hpp

5.983 std::chrono::tzdb_list Class Reference

`#include <chrono>`

Classes

- class const_iterator

Public Member Functions

- **tzdb_list** (const tzdb_list &)=delete
- const_iterator **begin** () const noexcept
- const_iterator **cbegin** () const noexcept
- const_iterator **cend** () const noexcept
- const_iterator **end** () const noexcept
- const_iterator erase_after (const_iterator __p)
- const tzdb & front () const noexcept
- tzdb_list & **operator=** (const tzdb_list &)=delete

Friends

- const tzdb & **get_tzdb** ()
- tzdb_list & **get_tzdb_list** ()
- class **leap_second**
- const tzdb & **reload_tzdb** ()
- struct **time_zone::_Impl**
- class **time_zone_link**
- struct **tzdb**

5.983.1 Detailed Description

The list of `chrono::tzdb` objects
A single object of this type is constructed by the C++ runtime, and can be accessed by calling `chrono::get_tzdb_list()`.
The front of the list is the current `tzdb` object and can be accessed via `chrono::get_tzdb_list().front()` or `chrono::get_tzdb()` or `*chronoget_tzdb_list().begin()`.
The `chrono::reload_tzdb()` function will check for a newer version and if found, insert it at the front of the list.

Since

C++20

5.983.2 Member Function Documentation

erase_after()

```
const_iterator std::chrono::tzdb_list::erase_after (
            const_iterator __p)
```

Remove the tzdb object *after* the one the iterator refers to.
Calling this function concurrently with any of `front()`, `begin()`, or `end()` does not cause a data race, but in general this function is not thread-safe. The behaviour may be undefined if erasing an element from the list while another thread is calling the same function, or incrementing an iterator into the list, or accessing the element being erased (unless it is accessed through an iterator).

Parameters

_↩ _p	A dereferenceable iterator.

Returns

An iterator the element after the one that was erased (or `end()` if there is no such element).

Since

C++20

front()

```
const tzdb & std::chrono::tzdb_list::front () const  [noexcept]
```

Access the current `tzdb` at the front of the list.
This returns a reference to the same object as `chrono::get_tzdb()`.

Returns

A reference to the current tzdb object.

Since

C++20

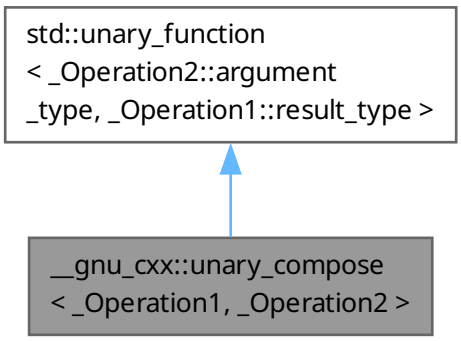
The documentation for this class was generated from the following file:

- chrono

5.984 __gnu_cxx::unary_compose< _Operation1, _Operation2 > Class Template Reference

```
#include <functional>
```

Inheritance diagram for __gnu_cxx::unary_compose< _Operation1, _Operation2 >:



Public Types

- typedef _Operation2::argument_type argument_type
- typedef _Operation1::result_type result_type

Public Member Functions

- **unary_compose** (const _Operation1 &__x, const _Operation2 &__y)
- _Operation1::result_type **operator()** (const typename _Operation2::argument_type &__x) const

Protected Attributes

- _Operation1 **_M_fn1**
- _Operation2 **_M_fn2**

5.984.1 Detailed Description

template<class _Operation1, class _Operation2>
class __gnu_cxx::unary_compose< _Operation1, _Operation2 >

An SGI extension .

5.984.2 Member Typedef Documentation

argument_type

```
typedef _Operation2::argument_type std::unary_function< _Operation2::argument_type, _Operation1↩
::result_type >::argument_type  [inherited]
```

`argument_type` is the type of the argument

result_type

```
typedef _Operation1::result_type std::unary_function< _Operation2::argument_type, _Operation1↩
::result_type >::result_type  [inherited]
```

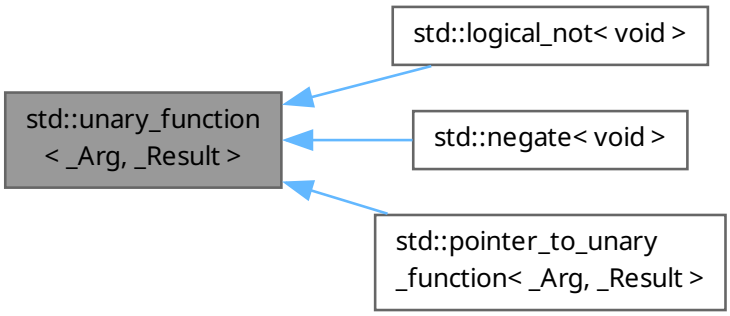
`result_type` is the return type

The documentation for this class was generated from the following file:

- ext/functional

5.985 std::unary_function< _Arg, _Result > Struct Template Reference

`#include <stl_function.h>`
Inheritance diagram for std::unary_function< _Arg, _Result >:



Public Types

- typedef _Arg argument_type
- typedef _Result result_type

5.985.1 Detailed Description

template<typename _Arg, typename _Result>
struct std::unary_function< _Arg, _Result >

Helper for defining adaptable unary function objects.

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

5.985.2 Member Typedef Documentation

argument_type

```
template<typename _Arg, typename _Result>
typedef _Arg std::unary_function< _Arg, _Result >::argument_type
```

`argument_type` is the type of the argument

result_type

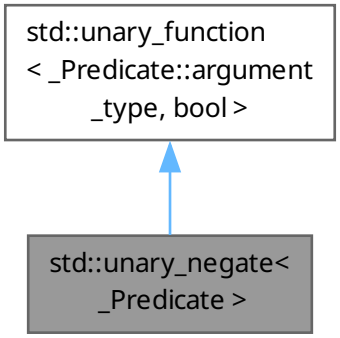
```
template<typename _Arg, typename _Result>
typedef _Result std::unary_function< _Arg, _Result >::result_type
```

`result_type` is the return type
The documentation for this struct was generated from the following file:

- stl_function.h

5.986 std::unary_negate< _Predicate > Class Template Reference

`#include <stl_function.h>`
Inheritance diagram for std::unary_negate< _Predicate >:



Public Types

- typedef _Predicate::argument_type argument_type
- typedef bool result_type

Public Member Functions

- constexpr **unary_negate** (const _Predicate &__x)
- constexpr bool **operator()** (const typename _Predicate::argument_type &__x) const

Protected Attributes

- _Predicate **_M_pred**

5.986.1 Detailed Description

template<**typename _Predicate**>
class std::unary_negate< **_Predicate** >

One of the negation functors.

5.986.2 Member Typedef Documentation

argument_type

`typedef _Predicate::argument_type std::unary_function< _Predicate::argument_type, bool >::argument_type [inherited]`
`argument_type` is the type of the argument

result_type

```
typedef bool std::unary_function< _Predicate::argument_type, bool >::result_type  [inherited]
```

result_type is the return type

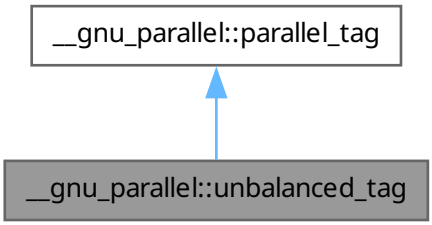
The documentation for this class was generated from the following file:

- stl_function.h

5.987 __gnu_parallel::unbalanced_tag Struct Reference

```
#include <tags.h>
```

Inheritance diagram for __gnu_parallel::unbalanced_tag:



Public Member Functions

- _ThreadIndex __get_num_threads ()
- void set_num_threads (_ThreadIndex __num_threads)

5.987.1 Detailed Description

Recommends parallel execution using static load-balancing at compile time.

5.987.2 Member Function Documentation

__get_num_threads()

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ()  [inline], [inherited]
```

Find out desired number of threads.

Returns

Desired number of threads.

Referenced by __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), and __gnu_parallel::__parallel_sort(

set_num_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
            _ThreadIndex __num_threads)  [inline], [inherited]
```

Set the desired number of threads.

Parameters

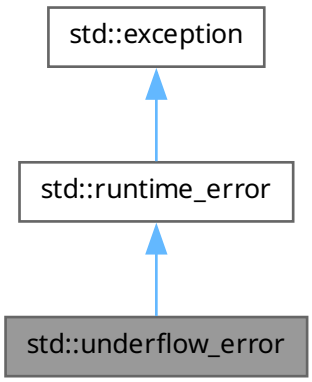
__num_threads	Desired number of threads.

The documentation for this struct was generated from the following file:

- tags.h

5.988 std::underflow_error Class Reference

`#include <stdexcept>`
Inheritance diagram for std::underflow_error:



Public Member Functions

- **underflow_error** (const char *)
- **underflow_error** (const string &__arg)
- **underflow_error** (const underflow_error &)=default
- **underflow_error** (underflow_error &&)=default
- underflow_error & **operator=** (const underflow_error &)=default
- underflow_error & **operator=** (underflow_error &&)=default
- virtual const char * what () const noexcept

5.988.1 Detailed Description

Thrown to indicate arithmetic underflow.

5.988.2 Member Function Documentation

what()

`virtual const char * std::runtime_error::what () const [virtual], [noexcept], [inherited]`
Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).
Reimplemented from std::exception.

Reimplemented in std::experimental::filesystem::v1::filesystem_error, and std::filesystem::filesystem_error.
The documentation for this class was generated from the following file:

- stdexcept

5.989 std::underlying_type< _Tp > Struct Template Reference

`#include <type_traits>`

5.989.1 Detailed Description

template<typename _Tp>
struct std::underlying_type< _Tp >

The underlying type of an enum.
The documentation for this struct was generated from the following file:

- type_traits

5.990 std::uniform_int_distribution< _IntType > Class Template Reference

`#include <random>`

Classes

- struct param_type

Public Types

- typedef _IntType result_type

Public Member Functions

- uniform_int_distribution ()
- uniform_int_distribution (_IntType __a, _IntType __b=__gnu_cxx::__int_traits< _IntType >::__max)
- **uniform_int_distribution** (const param_type &__p)
- template<typename _ForwardIterator, typename _UniformRandomBitGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomBitGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomBitGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomBitGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomBitGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomBitGenerator &__urng, const param_type &__p)
- result_type **a** () const
- result_type **b** () const
- result_type max () const
- result_type min () const
- template<typename _UniformRandomBitGenerator>
  result_type operator() (_UniformRandomBitGenerator &__urng)
- template<typename _UniformRandomBitGenerator>
  result_type **operator()** (_UniformRandomBitGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- bool operator== (const uniform_int_distribution &__d1, const uniform_int_distribution &__d2)

5.990.1 Detailed Description

template<typename _IntType = int>
class std::uniform_int_distribution< _IntType >

Uniform discrete distribution for random numbers. A discrete random distribution on the range $[min, max]$ with equal probability throughout the range.

Since

C++11

5.990.2 Member Typedef Documentation

result_type

```
template<typename _IntType = int>
typedef _IntType std::uniform_int_distribution< _IntType >::result_type
```

The type of the range of the distribution.

5.990.3 Constructor & Destructor Documentation

uniform_int_distribution() [1/2]

```
template<typename _IntType = int>
std::uniform_int_distribution< _IntType >::uniform_int_distribution ()  [inline]
```

Constructs a uniform distribution object.

uniform_int_distribution() [2/2]

```
template<typename _IntType = int>
std::uniform_int_distribution< _IntType >::uniform_int_distribution (
            _IntType __a,
            _IntType __b = __gnu_cxx::__int_traits<_IntType>::__max)  [inline], [explicit]
```

Constructs a uniform distribution object.

5.990.4 Member Function Documentation

max()

```
template<typename _IntType = int>
result_type std::uniform_int_distribution< _IntType >::max () const  [inline]
```

Returns the inclusive upper bound of the distribution range.

min()

```
template<typename _IntType = int>
result_type std::uniform_int_distribution< _IntType >::min () const  [inline]
```

Returns the inclusive lower bound of the distribution range.

operator()()

```
template<typename _IntType = int>
template<typename _UniformRandomBitGenerator>
```

```
result_type std::uniform_int_distribution< _IntType >::operator() (
            _UniformRandomBitGenerator & __urng)  [inline]
```

Generating functions.
Referenced by operator()().

param() [1/2]

```
template<typename _IntType = int>
param_type std::uniform_int_distribution< _IntType >::param () const  [inline]
```

Returns the parameter set of the distribution.
Referenced by std::operator>>().

param() [2/2]

```
template<typename _IntType = int>
void std::uniform_int_distribution< _IntType >::param (
            const param_type & __param)  [inline]
```

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

reset()

```
template<typename _IntType = int>
void std::uniform_int_distribution< _IntType >::reset ()  [inline]
```

Resets the distribution state.
Does nothing for the uniform integer distribution.

5.990.5 Friends And Related Symbol Documentation

operator==

```
template<typename _IntType = int>
bool operator== (
            const uniform_int_distribution< _IntType > & __d1,
            const uniform_int_distribution< _IntType > & __d2)  [friend]
```

Return true if two uniform integer distributions have the same parameters.
The documentation for this class was generated from the following file:

- uniform_int_dist.h

5.991 std::uniform_real_distribution< _RealType > Class Template Reference

```
#include <random>
```

Classes

- struct param_type

Public Types

- typedef _RealType result_type

Public Member Functions

- uniform_real_distribution ()
- uniform_real_distribution (_RealType __a, _RealType __b=_RealType(1))
- **uniform_real_distribution** (const param_type &__p)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- result_type **a** () const
- result_type **b** () const
- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- bool operator== (const uniform_real_distribution &__d1, const uniform_real_distribution &__d2)

5.991.1 Detailed Description

template<typename _RealType = double>
class std::uniform_real_distribution< _RealType >

Uniform continuous distribution for random numbers.
A continuous random distribution on the range [min, max) with equal probability throughout the range. The URNG should be real-valued and deliver number in the range [0, 1).

Since

C++11

5.991.2 Member Typedef Documentation

result_type

```
template<typename _RealType = double>
typedef _RealType std::uniform_real_distribution< _RealType >::result_type
```

The type of the range of the distribution.

5.991.3 Constructor & Destructor Documentation

uniform_real_distribution() [1/2]

```
template<typename _RealType = double>
std::uniform_real_distribution< _RealType >::uniform_real_distribution ()  [inline]
```

Constructs a uniform_real_distribution object.
The lower bound is set to 0.0 and the upper bound to 1.0

uniform_real_distribution() [2/2]

```
template<typename _RealType = double>
std::uniform_real_distribution< _RealType >::uniform_real_distribution (
            _RealType __a,
            _RealType __b = _RealType(1))  [inline], [explicit]
```

Constructs a uniform_real_distribution object.

Parameters

_↩ _a	[IN] The lower bound of the distribution.
_↩ _b	[IN] The upper bound of the distribution.

5.991.4 Member Function Documentation

max()

```
template<typename _RealType = double>
result_type std::uniform_real_distribution< _RealType >::max () const  [inline]
```

Returns the inclusive upper bound of the distribution range.

min()

```
template<typename _RealType = double>
result_type std::uniform_real_distribution< _RealType >::min () const  [inline]
```

Returns the inclusive lower bound of the distribution range.

operator()()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::uniform_real_distribution< _RealType >::operator() (
            _UniformRandomNumberGenerator & __urng)  [inline]
```

Generating functions.
Referenced by operator()().

param() [1/2]

```
template<typename _RealType = double>
param_type std::uniform_real_distribution< _RealType >::param () const  [inline]
```

Returns the parameter set of the distribution.
Referenced by std::operator>>().

param() [2/2]

```
template<typename _RealType = double>
void std::uniform_real_distribution< _RealType >::param (
            const param_type & __param)  [inline]
```

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

reset()

```
template<typename _RealType = double>
void std::uniform_real_distribution< _RealType >::reset ()  [inline]
```

Resets the distribution state.
Does nothing for the uniform real distribution.

5.991.5 Friends And Related Symbol Documentation

operator==

```
template<typename _RealType = double>
bool operator== (
            const uniform_real_distribution< _RealType > & __d1,
            const uniform_real_distribution< _RealType > & __d2)  [friend]
```

Return true if two uniform real distributions have the same parameters.
The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

5.992 std::unique_lock< _Mutex > Class Template Reference

```
#include <mutex>
```

Public Types

- typedef _Mutex **mutex_type**

Public Member Functions

- **unique_lock** (const unique_lock &)=delete
- **unique_lock** (mutex_type &__m)
- **unique_lock** (mutex_type &__m, adopt_lock_t) noexcept
- template<typename _Rep, typename _Period>
  unique_lock (mutex_type &__m, const chrono::duration< _Rep, _Period > &__rtime)
- template<typename _Clock, typename _Duration>
  unique_lock (mutex_type &__m, const chrono::time_point< _Clock, _Duration > &__atime)
- **unique_lock** (mutex_type &__m, defer_lock_t) noexcept
- **unique_lock** (mutex_type &__m, try_to_lock_t)
- **unique_lock** (unique_lock &&__u) noexcept
- void **lock** ()
- mutex_type * **mutex** () const noexcept
- **operator bool** () const noexcept
- unique_lock & **operator=** (const unique_lock &)=delete
- unique_lock & **operator=** (unique_lock &&__u) noexcept
- bool **owns_lock** () const noexcept
- mutex_type * **release** () noexcept
- void **swap** (unique_lock &__u) noexcept
- bool **try_lock** ()

- template<typename _Rep, typename _Period>
  bool **try_lock_for** (const chrono::duration< _Rep, _Period > &__rtime)
- template<typename _Clock, typename _Duration>
  bool **try_lock_until** (const chrono::time_point< _Clock, _Duration > &__atime)
- void **unlock** ()

Related Symbols

(Note that these are not member symbols.)

- template<typename _Mutex>
  void swap (unique_lock< _Mutex > &__x, unique_lock< _Mutex > &__y) noexcept

5.992.1 Detailed Description

template<typename _Mutex>
class std::unique_lock< _Mutex >

A movable scoped lock type.
A unique_lock controls mutex ownership within a scope. Ownership of the mutex can be delayed until after construction and can be transferred to another unique_lock by move construction or move assignment. If a mutex lock is owned when the destructor runs ownership will be released.

Since

C++11

5.992.2 Friends And Related Symbol Documentation

swap()

```
template<typename _Mutex>
void swap (
            unique_lock< _Mutex > & __x,
            unique_lock< _Mutex > & __y)  [related]
```

Swap overload for unique_lock objects.
The documentation for this class was generated from the following file:

- unique_lock.h

5.993 std::unique_ptr< _Tp, _Dp > Class Template Reference

```
#include <memory>
```

Public Types

- using **deleter_type**
- using **element_type**
- using **pointer**

Public Member Functions

- template<typename _Del = _Dp, typename = _DeleterConstraint<_Del>>
  constexpr unique_ptr () noexcept
- template<typename _Up, typename = _Require<is_convertible<_Up*, pointer>, is_same<_Dp, default_delete<_Tp>>>>
  unique_ptr (auto_ptr< _Up > &&__u) noexcept
- **unique_ptr** (const unique_ptr &)=delete

- template<typename _Del = _Dp, typename = _DeleterConstraint<_Del>>
  constexpr unique_ptr (nullptr_t) noexcept
- template<typename _Del = _Dp, typename = _DeleterConstraint<_Del>>
  constexpr unique_ptr (pointer __p) noexcept
- template<typename _Del = deleter_type, typename = _Require<is_move_constructible<_Del>>>
  constexpr unique_ptr (pointer __p, __enable_if_t<!is_lvalue_reference< _Del >::value, _Del && > __d) noexcept
- template<typename _Del = deleter_type, typename = _Require<is_copy_constructible<_Del>>>
  constexpr unique_ptr (pointer __p, const deleter_type &__d) noexcept
- template<typename _Del = deleter_type, typename _DelUnref = typename remove_reference<_Del>::type>
  constexpr **unique_ptr** (pointer, __enable_if_t< is_lvalue_reference< _Del >::value, _DelUnref && >)=delete
- unique_ptr (unique_ptr &&)=default
- template<typename _Up, typename _Ep, typename = _Require< __safe_conversion_up<_Up, _Ep>, __conditional_t<is_reference<_Dp>::value, is_same<_Ep, _Dp>, is_convertible<_Ep, _Dp>>>>
  constexpr unique_ptr (unique_ptr< _Up, _Ep > &&__u) noexcept
- ~unique_ptr () noexcept
- constexpr pointer get () const noexcept
- constexpr const deleter_type & get_deleter () const noexcept
- constexpr deleter_type & get_deleter () noexcept
- constexpr operator bool () const noexcept
- constexpr add_lvalue_reference< element_type >::type operator* () const noexcept(noexcept(*std::declval< pointer >()))
- constexpr pointer operator-> () const noexcept
- unique_ptr & **operator=** (const unique_ptr &)=delete
- constexpr unique_ptr & operator= (nullptr_t) noexcept
- unique_ptr & operator= (unique_ptr &&)=default
- template<typename _Up, typename _Ep>
  constexpr enable_if< __and_< __safe_conversion_up< _Up, _Ep >, is_assignable< deleter_type &, _Ep && > >::value, unique_ptr & >::type operator= (unique_ptr< _Up, _Ep > &&__u) noexcept
- constexpr pointer release () noexcept
- constexpr void reset (pointer __p=pointer()) noexcept
- constexpr void swap (unique_ptr &__u) noexcept

Related Symbols

(Note that these are not member symbols.)

- template<typename _CharT, typename _Traits, typename _Tp, typename _Dp>
  basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const unique_ptr< _Tp, _Dp > &__p)
- template<typename _Tp, typename _Dp>
  constexpr enable_if< __is_swappable< _Dp >::value >::type swap (unique_ptr< _Tp, _Dp > &__x, unique_ptr< _Tp, _Dp > &__y) noexcept

5.993.1 Detailed Description

template<typename _Tp, typename _Dp = default_delete<_Tp>>
class std::unique_ptr< _Tp, _Dp >

A move-only smart pointer that manages unique ownership of a resource.

Since

C++11

5.993.2 Constructor & Destructor Documentation

unique_ptr() [1/8]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
template<typename _Del = _Dp, typename = _DeleterConstraint<_Del>>
std::unique_ptr< _Tp, _Dp >::unique_ptr ()  [inline], [constexpr], [noexcept]
```

Default constructor, creates a unique_ptr that owns nothing.

unique_ptr() [2/8]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
template<typename _Del = _Dp, typename = _DeleterConstraint<_Del>>
std::unique_ptr< _Tp, _Dp >::unique_ptr (
            pointer __p)  [inline], [explicit], [constexpr], [noexcept]
```

Takes ownership of a pointer.

Parameters

_↩ _p	A pointer to an object of `element_type`

The deleter will be value-initialized.

unique_ptr() [3/8]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
template<typename _Del = deleter_type, typename = _Require<is_copy_constructible<_Del>>>
std::unique_ptr< _Tp, _Dp >::unique_ptr (
            pointer __p,
            const deleter_type & __d)  [inline], [constexpr], [noexcept]
```

Takes ownership of a pointer.

Parameters

_↩ _p	A pointer to an object of `element_type`
_↩ _d	A reference to a deleter.

The deleter will be initialized with `__d`

unique_ptr() [4/8]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
template<typename _Del = deleter_type, typename = _Require<is_move_constructible<_Del>>>
std::unique_ptr< _Tp, _Dp >::unique_ptr (
            pointer __p,
            __enable_if_t<!is_lvalue_reference< _Del >::value, _Del && > __d)  [inline], [constexpr],
[noexcept]
```

Takes ownership of a pointer.

Parameters

_↩ _p	A pointer to an object of element_type
_↩ _d	An rvalue reference to a (non-reference) deleter.

The deleter will be initialized with std::move(__d)

unique_ptr() [5/8]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
template<typename _Del = _Dp, typename = _DeleterConstraint<_Del>>
std::unique_ptr< _Tp, _Dp >::unique_ptr (
            nullptr_t )  [inline], [constexpr], [noexcept]
```

Creates a unique_ptr that owns nothing.

unique_ptr() [6/8]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
std::unique_ptr< _Tp, _Dp >::unique_ptr (
            unique_ptr< _Tp, _Dp > && )  [default]
```

Move constructor.

unique_ptr() [7/8]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
template<typename _Up, typename _Ep, typename = _Require< __safe_conversion_up<_Up, _Ep>, __↩
conditional_t<is_reference<_Dp>::value, is_same<_Ep, _Dp>, is_convertible<_Ep, _Dp>>>>
std::unique_ptr< _Tp, _Dp >::unique_ptr (
            unique_ptr< _Up, _Ep > && __u)  [inline], [constexpr], [noexcept]
```

Converting constructor from another type.
Requires that the pointer owned by __u is convertible to the type of pointer owned by this object, __u does not own an array, and __u has a compatible deleter type.

unique_ptr() [8/8]

```
template<typename _Tp, typename _Dp>
template<typename _Up, typename>
std::unique_ptr< _Tp, _Dp >::unique_ptr (
            auto_ptr< _Up > && __u)  [inline], [noexcept]
```

Converting constructor from auto_ptr.

~unique_ptr()

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
std::unique_ptr< _Tp, _Dp >::~unique_ptr ()  [inline], [noexcept]
```

Destructor, invokes the deleter if the stored pointer is not null.

5.993.3 Member Function Documentation

get()

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
pointer std::unique_ptr< _Tp, _Dp >::get () const  [inline], [constexpr], [noexcept]
```

Return the stored pointer.
Referenced by std::unique_ptr< _State >::operator bool(), std::unique_ptr< _State >::operator*(), std::unique_ptr< _State >::operator-> std::operator<(), std::operator<(), std::operator<(), std::unique_ptr< _State >::operator<<(), std::operator==(), std::operator>(), and std::operator>().

get_deleter() [1/2]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
const deleter_type & std::unique_ptr< _Tp, _Dp >::get_deleter () const  [inline], [constexpr],
[noexcept]
```

Return a reference to the stored deleter.

get_deleter() [2/2]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
deleter_type & std::unique_ptr< _Tp, _Dp >::get_deleter ()  [inline], [constexpr], [noexcept]
```

Return a reference to the stored deleter.
Referenced by std::unique_ptr< _State >::~unique_ptr(), and std::unique_ptr< _State >::operator=().

operator bool()

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
std::unique_ptr< _Tp, _Dp >::operator bool () const  [inline], [explicit], [constexpr], [noexcept]
```

Return `true` if the stored pointer is not null.

operator*()

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
add_lvalue_reference< element_type >::type std::unique_ptr< _Tp, _Dp >::operator* () const  [inline],
[constexpr], [noexcept]
```

Dereference the stored pointer.

operator->()

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
pointer std::unique_ptr< _Tp, _Dp >::operator-> () const  [inline], [constexpr], [noexcept]
```

Return the stored pointer.

operator=() [1/3]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
unique_ptr & std::unique_ptr< _Tp, _Dp >::operator= (
            nullptr_t )  [inline], [constexpr], [noexcept]
```

Reset the unique_ptr to empty, invoking the deleter if necessary.

operator=() [2/3]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
unique_ptr & std::unique_ptr< _Tp, _Dp >::operator= (
            unique_ptr< _Tp, _Dp > && )  [default]
```

Move assignment operator.
Invokes the deleter if this object owns a pointer.

operator=() [3/3]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
template<typename _Up, typename _Ep>
enable_if< __and_< __safe_conversion_up< _Up, _Ep >, is_assignable< deleter_type &, _Ep && >
>::value, unique_ptr & >::type std::unique_ptr< _Tp, _Dp >::operator= (
            unique_ptr< _Up, _Ep > && __u)  [inline], [constexpr], [noexcept]
```

Assignment from another type.

Parameters

_↩ _u	The object to transfer ownership from, which owns a convertible pointer to a non-array object.

Invokes the deleter if this object owns a pointer.

release()

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
pointer std::unique_ptr< _Tp, _Dp >::release ()  [inline], [constexpr], [noexcept]
```

Release ownership of any stored pointer.

reset()

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
void std::unique_ptr< _Tp, _Dp >::reset (
            pointer __p = pointer())  [inline], [constexpr], [noexcept]
```

Replace the stored pointer.

Parameters

_↩ _p	The new pointer to store.

The deleter will be invoked if a pointer is already owned.
Referenced by std::unique_ptr< _State >::operator=(), and std::unique_ptr< _State >::operator=().

swap()

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
void std::unique_ptr< _Tp, _Dp >::swap (
            unique_ptr< _Tp, _Dp > & __u)  [inline], [constexpr], [noexcept]
```

Exchange the pointer and deleter with another object.
The documentation for this class was generated from the following files:

- unique_ptr.h
- auto_ptr.h

5.994 std::unique_ptr< _Tp[], _Dp > Class Template Reference

```
#include <memory>
```

Public Types

- template<typename _Up>
  using **__safe_conversion_raw**
- template<typename _Up, typename _Ep, typename _UPtr = unique_ptr<_Up, _Ep>, typename _UP_pointer = typename _UPtr::pointer, typename _UP_element_type = typename _UPtr::element_type>
  using **__safe_conversion_up**
- using **deleter_type**
- using **deleter_type**
- using **element_type**
- using **element_type**
- using **pointer**
- using **pointer**

Public Member Functions

- constexpr unique_ptr () noexcept
- template<typename _Del = _Dp, typename = _DeleterConstraint<_Del>>
  constexpr unique_ptr () noexcept
- template<typename _Up, typename _Vp = _Dp, typename = _DeleterConstraint<_Vp>, typename = typename enable_if< __safe_conversion_raw<_Up>::value, bool>::type>
  constexpr unique_ptr (_Up __p) noexcept
- template<typename _Up, typename _Del = deleter_type, typename = _Require<__safe_conversion_raw<_Up>, is_move_constructible<_Del>>>
  constexpr unique_ptr (_Up __p, __enable_if_t<!is_lvalue_reference< _Del >::value, _Del && > __d) noexcept
- template<typename _Up, typename _Del = deleter_type, typename = _Require<__safe_conversion_raw<_Up>, is_copy_constructible<_Del>>>
  constexpr unique_ptr (_Up __p, const deleter_type &__d) noexcept
- template<typename _Up, typename _Del = deleter_type, typename _DelUnref = typename remove_reference<_Del>::type, typename = _Require<__safe_conversion_raw<_Up>>>
  unique_ptr (_Up, __enable_if_t< is_lvalue_reference< _Del >::value, _DelUnref && >)=delete
- unique_ptr (auto_ptr< _Up > &&__u) noexcept
- **unique_ptr** (const unique_ptr &)=delete
- **unique_ptr** (const unique_ptr &)=delete
- constexpr unique_ptr (nullptr_t) noexcept
- template<typename _Del = _Dp, typename = _DeleterConstraint<_Del>>
  constexpr unique_ptr (nullptr_t) noexcept
- constexpr unique_ptr (pointer __p) noexcept
- constexpr unique_ptr (pointer __p, __enable_if_t<!is_lvalue_reference< _Del >::value, _Del && > __d) noexcept
- constexpr unique_ptr (pointer __p, const deleter_type &__d) noexcept
- constexpr **unique_ptr** (pointer, __enable_if_t< is_lvalue_reference< _Del >::value, _DelUnref && >)=delete
- unique_ptr (unique_ptr &&)=default
- unique_ptr (unique_ptr &&)=default
- constexpr unique_ptr (unique_ptr< _Up, _Ep > &&__u) noexcept
- template<typename _Up, typename _Ep, typename = _Require< __safe_conversion_up<_Up, _Ep>, __conditional_t<is_reference<_Dp>::value, is_same<_Ep, _Dp>, is_convertible<_Ep, _Dp>>>>
  constexpr **unique_ptr** (unique_ptr< _Up, _Ep > &&__u) noexcept
- ~unique_ptr ()
- ~unique_ptr () noexcept
- constexpr pointer get () const noexcept
- constexpr pointer get () const noexcept
- constexpr const deleter_type & get_deleter () const noexcept

- constexpr const deleter_type & get_deleter () const noexcept
- constexpr deleter_type & get_deleter () noexcept
- constexpr deleter_type & get_deleter () noexcept
- constexpr operator bool () const noexcept
- constexpr operator bool () const noexcept
- constexpr add_lvalue_reference< element_type >::type operator* () const noexcept(noexcept(*std::declval< pointer >()))
- constexpr pointer operator-> () const noexcept
- unique_ptr & **operator=** (const unique_ptr &)=delete
- unique_ptr & **operator=** (const unique_ptr &)=delete
- constexpr unique_ptr & operator= (nullptr_t) noexcept
- constexpr unique_ptr & operator= (nullptr_t) noexcept
- unique_ptr & operator= (unique_ptr &&)=default
- unique_ptr & operator= (unique_ptr &&)=default
- constexpr enable_if< __and_< __safe_conversion_up< _Up, _Ep >, is_assignable< deleter_type &, _Ep && > >::value, unique_ptr & >::type operator= (unique_ptr< _Up, _Ep > &&__u) noexcept
- template<typename _Up, typename _Ep>
  constexpr enable_if< __and_< __safe_conversion_up< _Up, _Ep >, is_assignable< deleter_type &, _Ep && > >::value, unique_ptr & >::type operator= (unique_ptr< _Up, _Ep > &&__u) noexcept
- constexpr std::add_lvalue_reference< element_type >::type operator[ ] (size_t __i) const
- constexpr pointer release () noexcept
- constexpr pointer release () noexcept
- template<typename _Up, typename = _Require< __or_<is_same<_Up, pointer>, __and_<is_same<pointer, element_type*>, is_pointer<_Up>, is_convertible< typename remove_pointer<_Up>::type(*)[ ], element_type(*)[ ] > > > >>
  constexpr void reset (_Up __p) noexcept
- constexpr void **reset** (nullptr_t=nullptr) noexcept
- constexpr void reset (pointer __p=pointer()) noexcept
- constexpr void swap (unique_ptr &__u) noexcept
- constexpr void swap (unique_ptr &__u) noexcept

Related Symbols

(Note that these are not member symbols.)

- basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const unique_ptr< _Tp, _Dp > &__p)
- basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const unique_ptr< _Tp, _Dp > &__p)
- constexpr enable_if< __is_swappable< _Dp >::value >::type swap (unique_ptr< _Tp, _Dp > &__x, unique_ptr< _Tp, _Dp > &__y) noexcept
- constexpr enable_if< __is_swappable< _Dp >::value >::type swap (unique_ptr< _Tp, _Dp > &__x, unique_ptr< _Tp, _Dp > &__y) noexcept

5.994.1 Detailed Description

template<typename _Tp, typename _Dp>
class std::unique_ptr< _Tp[ ], _Dp >

A move-only smart pointer that manages unique ownership of an array.

Since

C++11

5.994.2 Constructor & Destructor Documentation

unique_ptr() [1/14]

```
template<typename _Tp, typename _Dp>
template<typename _Del = _Dp, typename = _DeleterConstraint<_Del>>
std::unique_ptr< _Tp[], _Dp >::unique_ptr ()  [inline], [constexpr], [noexcept]
```

Default constructor, creates a unique_ptr that owns nothing.
References unique_ptr().
Referenced by unique_ptr(), unique_ptr(), unique_ptr(), unique_ptr(), unique_ptr(), unique_ptr(), ~unique_ptr(), operator=(), operator=(), operator=(), and swap().

unique_ptr() [2/14]

```
template<typename _Tp, typename _Dp>
template<typename _Up, typename _Vp = _Dp, typename = _DeleterConstraint<_Vp>, typename = typename
enable_if< __safe_conversion_raw<_Up>::value, bool>::type>
std::unique_ptr< _Tp[], _Dp >::unique_ptr (
            _Up __p)  [inline], [explicit], [constexpr], [noexcept]
```

Takes ownership of a pointer.

Parameters

_↩ _p	A pointer to an array of a type safely convertible to an array of element_type

The deleter will be value-initialized.
References unique_ptr().

unique_ptr() [3/14]

```
template<typename _Tp, typename _Dp>
template<typename _Up, typename _Del = deleter_type, typename = _Require<__safe_conversion_raw<↩
_Up>, is_copy_constructible<_Del>>>
std::unique_ptr< _Tp[], _Dp >::unique_ptr (
            _Up __p,
            const deleter_type & __d)  [inline], [constexpr], [noexcept]
```

Takes ownership of a pointer.

Parameters

_↩ _p	A pointer to an array of a type safely convertible to an array of element_type
_↩ _d	A reference to a deleter.

The deleter will be initialized with __d
References unique_ptr().

unique_ptr() [4/14]

```
template<typename _Tp, typename _Dp>
template<typename _Up, typename _Del = deleter_type, typename = _Require<__safe_conversion_raw<↩
_Up>, is_move_constructible<_Del>>>
std::unique_ptr< _Tp[], _Dp >::unique_ptr (
```

```
            _Up __p,
            __enable_if_t<!is_lvalue_reference< _Del >::value, _Del && > __d)  [inline], [constexpr],
[noexcept]
```

Takes ownership of a pointer.

Parameters

_↩ _p	A pointer to an array of a type safely convertible to an array of `element_type`
_↩ _d	A reference to a deleter.

The deleter will be initialized with `std::move(__d)`
References unique_ptr(), and std::move().

unique_ptr() [5/14]

```
template<typename _Tp, typename _Dp>
std::unique_ptr< _Tp[], _Dp >::unique_ptr (
            unique_ptr< _Tp[], _Dp > && )  [default]
```

Move constructor.
References unique_ptr().

unique_ptr() [6/14]

```
template<typename _Tp, typename _Dp>
template<typename _Del = _Dp, typename = _DeleterConstraint<_Del>>
std::unique_ptr< _Tp[], _Dp >::unique_ptr (
            nullptr_t )  [inline], [constexpr], [noexcept]
```

Creates a unique_ptr that owns nothing.
References unique_ptr().

~unique_ptr() [1/2]

```
template<typename _Tp, typename _Dp>
std::unique_ptr< _Tp[], _Dp >::~unique_ptr ()  [inline]
```

Destructor, invokes the deleter if the stored pointer is not null.
References unique_ptr(), and get_deleter().

unique_ptr() [7/14]

```
std::unique_ptr< _Tp, _Dp >::unique_ptr ()  [inline], [constexpr], [noexcept]
```

Default constructor, creates a unique_ptr that owns nothing.

unique_ptr() [8/14]

```
std::unique_ptr< _Tp, _Dp >::unique_ptr (
            pointer __p)  [inline], [explicit], [constexpr], [noexcept]
```

Takes ownership of a pointer.

Parameters

_↩ _p	A pointer to an object of `element_type`

The deleter will be value-initialized.

unique_ptr() [9/14]

```
std::unique_ptr< _Tp, _Dp >::unique_ptr (
            pointer __p,
            const deleter_type & __d)  [inline], [constexpr], [noexcept]
```

Takes ownership of a pointer.

Parameters

_↩ _p	A pointer to an object of element_type
_↩ _d	A reference to a deleter.

The deleter will be initialized with __d

unique_ptr() [10/14]

```
std::unique_ptr< _Tp, _Dp >::unique_ptr (
            pointer __p,
            __enable_if_t<!is_lvalue_reference< _Del >::value, _Del && > __d)  [inline], [constexpr],
[noexcept]
```

Takes ownership of a pointer.

Parameters

_↩ _p	A pointer to an object of element_type
_↩ _d	An rvalue reference to a (non-reference) deleter.

The deleter will be initialized with std::move(__d)

unique_ptr() [11/14]

```
std::unique_ptr< _Tp, _Dp >::unique_ptr (
            nullptr_t )  [inline], [constexpr], [noexcept]
```

Creates a unique_ptr that owns nothing.

unique_ptr() [12/14]

```
std::unique_ptr< _Tp, _Dp >::unique_ptr (
            unique_ptr< _Tp[], _Dp > && )  [default]
```

Move constructor.

unique_ptr() [13/14]

```
std::unique_ptr< _Tp, _Dp >::unique_ptr (
            unique_ptr< _Up, _Ep > && __u)  [inline], [constexpr], [noexcept]
```

Converting constructor from another type.
Requires that the pointer owned by __u is convertible to the type of pointer owned by this object, __u does not own an array, and __u has a compatible deleter type.

unique_ptr() [14/14]

```
std::unique_ptr< _Tp, _Dp >::unique_ptr (
            auto_ptr< _Up > && __u)  [inline], [noexcept]
```

Converting constructor from `auto_ptr`.

~unique_ptr() [2/2]

```
std::unique_ptr< _Tp, _Dp >::~unique_ptr ()  [inline], [noexcept]
```

Destructor, invokes the deleter if the stored pointer is not null.

5.994.3 Member Function Documentation

get() [1/2]

```
pointer std::unique_ptr< _Tp, _Dp >::get () const  [inline], [constexpr], [noexcept]
```

Return the stored pointer.

get() [2/2]

```
template<typename _Tp, typename _Dp>
pointer std::unique_ptr< _Tp[], _Dp >::get () const  [inline], [constexpr], [noexcept]
```

Return the stored pointer.
References get().
Referenced by get(), operator bool(), and operator[]().

get_deleter() [1/4]

```
const deleter_type & std::unique_ptr< _Tp, _Dp >::get_deleter () const  [inline], [constexpr],
[noexcept]
```

Return a reference to the stored deleter.

get_deleter() [2/4]

```
template<typename _Tp, typename _Dp>
const deleter_type & std::unique_ptr< _Tp[], _Dp >::get_deleter () const  [inline], [constexpr],
[noexcept]
```

Return a reference to the stored deleter.
References get_deleter().

get_deleter() [3/4]

```
deleter_type & std::unique_ptr< _Tp, _Dp >::get_deleter ()  [inline], [constexpr], [noexcept]
```

Return a reference to the stored deleter.

get_deleter() [4/4]

```
template<typename _Tp, typename _Dp>
deleter_type & std::unique_ptr< _Tp[], _Dp >::get_deleter ()  [inline], [constexpr], [noexcept]
```

Return a reference to the stored deleter.
References get_deleter().
Referenced by ~unique_ptr(), get_deleter(), get_deleter(), and operator=().

operator bool() [1/2]

```
std::unique_ptr< _Tp, _Dp >::operator bool () const  [inline], [explicit], [constexpr], [noexcept]
```

Return `true` if the stored pointer is not null.

operator bool() [2/2]

```
template<typename _Tp, typename _Dp>
std::unique_ptr< _Tp[], _Dp >::operator bool () const  [inline], [explicit], [constexpr], [noexcept]
```

Return `true` if the stored pointer is not null.
References get().

operator*()

```
add_lvalue_reference< element_type >::type std::unique_ptr< _Tp, _Dp >::operator* () const  [inline],
[constexpr], [noexcept]
```

Dereference the stored pointer.

operator->()

```
pointer std::unique_ptr< _Tp, _Dp >::operator-> () const  [inline], [constexpr], [noexcept]
```

Return the stored pointer.

operator=() [1/6]

```
unique_ptr & std::unique_ptr< _Tp, _Dp >::operator= (
            nullptr_t )  [inline], [constexpr], [noexcept]
```

Reset the unique_ptr to empty, invoking the deleter if necessary.

operator=() [2/6]

```
template<typename _Tp, typename _Dp>
unique_ptr & std::unique_ptr< _Tp[], _Dp >::operator= (
            nullptr_t )  [inline], [constexpr], [noexcept]
```

Reset the unique_ptr to empty, invoking the deleter if necessary.
References unique_ptr(), operator=(), and reset().

operator=() [3/6]

```
unique_ptr & std::unique_ptr< _Tp, _Dp >::operator= (
            unique_ptr< _Tp[], _Dp > && )  [default]
```

Move assignment operator.
Invokes the deleter if this object owns a pointer.

operator=() [4/6]

```
template<typename _Tp, typename _Dp>
unique_ptr & std::unique_ptr< _Tp[], _Dp >::operator= (
            unique_ptr< _Tp[], _Dp > && )  [default]
```

Move assignment operator.
Invokes the deleter if this object owns a pointer.
References unique_ptr(), and operator=().
Referenced by operator=(), operator=(), and operator=().

operator=() [5/6]

```
enable_if< __and_< __safe_conversion_up< _Up, _Ep >, is_assignable< deleter_type &, _Ep && >
>::value, unique_ptr & >::type std::unique_ptr< _Tp, _Dp >::operator= (
            unique_ptr< _Up, _Ep > && __u)  [inline], [constexpr], [noexcept]
```

Assignment from another type.

Parameters

_↩ _u	The object to transfer ownership from, which owns a convertible pointer to a non-array object.

Invokes the deleter if this object owns a pointer.

operator=() [6/6]

```
template<typename _Tp, typename _Dp>
template<typename _Up, typename _Ep>
enable_if< __and_< __safe_conversion_up< _Up, _Ep >, is_assignable< deleter_type &, _Ep && >
>::value, unique_ptr & >::type std::unique_ptr< _Tp[], _Dp >::operator= (
            unique_ptr< _Up, _Ep > && __u)  [inline], [constexpr], [noexcept]
```

Assignment from another type.

Parameters

_↩ _u	The object to transfer ownership from, which owns a convertible pointer to an array object.

Invokes the deleter if this object owns a pointer.
References unique_ptr(), std::forward(), get_deleter(), operator=(), and reset().

operator[]()

```
template<typename _Tp, typename _Dp>
std::add_lvalue_reference< element_type >::type std::unique_ptr< _Tp[], _Dp >::operator[] (
            size_t __i) const  [inline], [constexpr]
```

Access an element of owned array.
References get(), and operator[ ]().
Referenced by operator[ ]().

release() [1/2]

```
pointer std::unique_ptr< _Tp, _Dp >::release ()  [inline], [constexpr], [noexcept]
```

Release ownership of any stored pointer.

release() [2/2]

```
template<typename _Tp, typename _Dp>
pointer std::unique_ptr< _Tp[], _Dp >::release ()  [inline], [constexpr], [noexcept]
```

Release ownership of any stored pointer.
References release().
Referenced by release().

reset() [1/2]

```
template<typename _Tp, typename _Dp>
template<typename _Up, typename = _Require< __or_<is_same<_Up, pointer>, __and_<is_same<pointer,
element_type*>, is_pointer<_Up>, is_convertible< typename remove_pointer<_Up>::type(*)[], element↩
_type(*)[] > > > >>
void std::unique_ptr< _Tp[], _Dp >::reset (
            _Up __p)  [inline], [constexpr], [noexcept]
```

Replace the stored pointer.

Parameters

_↩ _p	The new pointer to store.

The deleter will be invoked if a pointer is already owned.
References std::move(), and reset().
Referenced by operator=(), operator=(), and reset().

reset() [2/2]

```
void std::unique_ptr< _Tp, _Dp >::reset (
            pointer __p = pointer())  [inline], [constexpr], [noexcept]
```

Replace the stored pointer.

Parameters

_↩ _p	The new pointer to store.

The deleter will be invoked if a pointer is already owned.

swap() [1/2]

```
void std::unique_ptr< _Tp, _Dp >::swap (
            unique_ptr< _Tp[], _Dp > & __u)  [inline], [constexpr], [noexcept]
```

Exchange the pointer and deleter with another object.

swap() [2/2]

```
template<typename _Tp, typename _Dp>
void std::unique_ptr< _Tp[], _Dp >::swap (
            unique_ptr< _Tp[], _Dp > & __u)  [inline], [constexpr], [noexcept]
```

Exchange the pointer and deleter with another object.
References unique_ptr(), and swap().
Referenced by swap().

5.994.4 Friends And Related Symbol Documentation

operator<<() [1/2]

```
basic_ostream< _CharT, _Traits > & operator<< (
            basic_ostream< _CharT, _Traits > & __os,
            const unique_ptr< _Tp, _Dp > & __p)  [related]
```

Stream output operator for unique_ptr.

Since

C++20

operator<<() [2/2]

```
basic_ostream< _CharT, _Traits > & operator<< (
            basic_ostream< _CharT, _Traits > & __os,
            const unique_ptr< _Tp, _Dp > & __p)  [related]
```

Stream output operator for unique_ptr.

Since

C++20

swap() [1/2]

```
enable_if< __is_swappable< _Dp >::value >::type swap (
            unique_ptr< _Tp, _Dp > & __x,
            unique_ptr< _Tp, _Dp > & __y)  [related]
```

Swap overload for unique_ptr.

swap() [2/2]

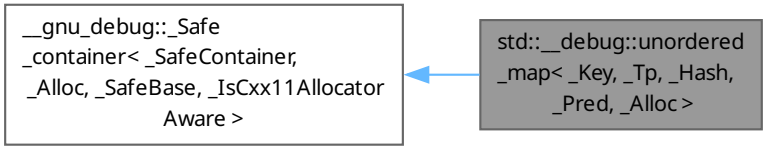
```
enable_if< __is_swappable< _Dp >::value >::type swap (
            unique_ptr< _Tp, _Dp > & __x,
            unique_ptr< _Tp, _Dp > & __y)  [related]
```

Swap overload for unique_ptr.
The documentation for this class was generated from the following files:

- unique_ptr.h
- auto_ptr.h

5.995 std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference

`#include <unordered_map>`
Inheritance diagram for std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >:



Public Types

- typedef _Base::allocator_type **allocator_type**
- typedef __gnu_debug::_Safe_iterator< _Base_const_iterator, unordered_map > **const_iterator**
- typedef __gnu_debug::_Safe_local_iterator< _Base_const_local_iterator, unordered_map > **const_local_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**
- typedef _Base::difference_type **difference_type**
- typedef _Base::hasher **hasher**
- using **insert_return_type**
- typedef __gnu_debug::_Safe_iterator< _Base_iterator, unordered_map > **iterator**
- typedef _Base::key_equal **key_equal**
- typedef _Base::key_type **key_type**

- typedef __gnu_debug::_Safe_local_iterator< _Base_local_iterator, unordered_map > **local_iterator**
- typedef _Base::mapped_type **mapped_type**
- using **node_type**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef _Base::size_type **size_type**
- typedef _Base::value_type **value_type**

Public Member Functions

- **unordered_map** (_Base_ref __x)
- template<typename _InputIterator>
  unordered_map (_InputIterator __first, _InputIterator __last, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_map (_InputIterator __first, _InputIterator __last, size_type __n, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_map (_InputIterator __first, _InputIterator __last, size_type __n, const hasher &__hf, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_map (_InputIterator __first, _InputIterator __last, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_map** (const allocator_type &__a)
- **unordered_map** (const unordered_map &)=default
- **unordered_map** (const unordered_map &__umap, const allocator_type &__a)
- **unordered_map** (initializer_list< value_type > __l, const allocator_type &__a)
- **unordered_map** (initializer_list< value_type > __l, size_type __n, const allocator_type &__a)
- **unordered_map** (initializer_list< value_type > __l, size_type __n, const hasher &__hf, const allocator_type &__a)
- **unordered_map** (initializer_list< value_type > __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_map** (size_type __n, const allocator_type &__a)
- **unordered_map** (size_type __n, const hasher &__hf, const allocator_type &__a)
- **unordered_map** (size_type __n, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_map** (unordered_map &&)=default
- **unordered_map** (unordered_map &&__umap, const allocator_type &__a) noexcept(noexcept(_Base(std::move(__umap), __a)))
- const _Base & **_M_base** () const noexcept
- _Base & **_M_base** () noexcept
- const_iterator **begin** () const noexcept
- iterator **begin** () noexcept
- local_iterator **begin** (size_type __b)
- const_local_iterator **begin** (size_type __b) const
- size_type **bucket_size** (size_type __b) const
- const_iterator **cbegin** () const noexcept
- const_local_iterator **cbegin** (size_type __b) const
- const_iterator **cend** () const noexcept
- const_local_iterator **cend** (size_type __b) const
- void **clear** () noexcept
- template<typename... _Args>
  std::pair< iterator, bool > **emplace** (_Args &&... __args)

- template<typename... _Args>
  iterator **emplace_hint** (const_iterator __hint, _Args &&... __args)
- const_iterator **end** () const noexcept
- iterator **end** () noexcept
- local_iterator **end** (size_type __b)
- const_local_iterator **end** (size_type __b) const
- template<typename _Kt, typename = std::__has_is_transparent_t<_Hash, _Kt>, typename = std::__has_is_transparent_t<_Pred, _Kt>>
  std::pair< iterator, iterator > **equal_range** (const _Kt &__k)
- template<typename _Kt, typename = std::__has_is_transparent_t<_Hash, _Kt>, typename = std::__has_is_transparent_t<_Pred, _Kt>>
  std::pair< const_iterator, const_iterator > **equal_range** (const _Kt &__k) const
- std::pair< iterator, iterator > **equal_range** (const key_type &__key)
- std::pair< const_iterator, const_iterator > **equal_range** (const key_type &__key) const
- _Base_iterator **erase** (_Base_const_iterator __it)
- size_type **erase** (const key_type &__key)
- iterator **erase** (const_iterator __first, const_iterator __last)
- iterator **erase** (const_iterator __it)
- iterator **erase** (iterator __it)
- node_type **extract** (const key_type &__key)
- node_type **extract** (const_iterator __position)
- template<typename _Kt, typename = std::__has_is_transparent_t<_Hash, _Kt>, typename = std::__has_is_transparent_t<_Pred, _Kt>>
  iterator **find** (const _Kt &__k)
- template<typename _Kt, typename = std::__has_is_transparent_t<_Hash, _Kt>, typename = std::__has_is_transparent_t<_Pred, _Kt>>
  const_iterator **find** (const _Kt &__k) const
- iterator **find** (const key_type &__key)
- const_iterator **find** (const key_type &__key) const
- template<typename _InputIterator>
  void **insert** (_InputIterator __first, _InputIterator __last)
- template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
  std::pair< iterator, bool > **insert** (_Pair &&__obj)
- std::pair< iterator, bool > **insert** (const value_type &__obj)
- template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
  iterator **insert** (const_iterator __hint, _Pair &&__obj)
- iterator **insert** (const_iterator __hint, const value_type &__obj)
- iterator **insert** (const_iterator __hint, node_type &&__nh)
- iterator **insert** (const_iterator __hint, value_type &&__x)
- insert_return_type **insert** (node_type &&__nh)
- void **insert** (std::initializer_list< value_type > __l)
- std::pair< iterator, bool > **insert** (value_type &&__x)
- float **max_load_factor** () const noexcept
- void **max_load_factor** (float __f)
- template<typename _H2, typename _P2>
  void **merge** (unordered_map< _Key, _Tp, _H2, _P2, _Alloc > &&__source)
- template<typename _H2, typename _P2>
  void **merge** (unordered_map< _Key, _Tp, _H2, _P2, _Alloc > &__source)
- template<typename _H2, typename _P2>
  void **merge** (unordered_multimap< _Key, _Tp, _H2, _P2, _Alloc > &&__source)
- template<typename _H2, typename _P2>
  void **merge** (unordered_multimap< _Key, _Tp, _H2, _P2, _Alloc > &__source)
- unordered_map & **operator=** (const unordered_map &)=default
- unordered_map & **operator=** (initializer_list< value_type > __l)
- unordered_map & **operator=** (unordered_map &&)=default
- void **swap** (unordered_map &__x) noexcept(noexcept(declval< _Base & >().swap(__x)))

Protected Member Functions

- constexpr void **_M_swap** (const _Safe_container &__x) const noexcept

Friends

- template<typename _ItT, typename _SeqT, typename _CatT>
  class **::__gnu_debug::_Safe_iterator**
- template<typename _ItT, typename _SeqT>
  class **::__gnu_debug::_Safe_local_iterator**

5.995.1 Detailed Description

template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> > >
class std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >

Class std::unordered_map with safety/checking/debug instrumentation.
The documentation for this class was generated from the following file:

- debug/unordered_map

5.996 std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference

`#include <unordered_map>`

Public Types

- typedef _Hashtable::key_type key_type
- typedef _Hashtable::value_type value_type
- typedef _Hashtable::mapped_type mapped_type
- typedef _Hashtable::hasher hasher
- typedef _Hashtable::key_equal key_equal
- typedef _Hashtable::allocator_type allocator_type

- typedef _Hashtable::pointer pointer
- typedef _Hashtable::const_pointer const_pointer
- typedef _Hashtable::reference reference
- typedef _Hashtable::const_reference const_reference
- typedef _Hashtable::iterator iterator
- typedef _Hashtable::const_iterator const_iterator
- typedef _Hashtable::local_iterator local_iterator
- typedef _Hashtable::const_local_iterator const_local_iterator
- typedef _Hashtable::size_type size_type
- typedef _Hashtable::difference_type difference_type

Public Member Functions

- unordered_map ()=default
- template<typename _InputIterator>
  unordered_map (_InputIterator __first, _InputIterator __last, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_map (_InputIterator __first, _InputIterator __last, size_type __n, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_map (_InputIterator __first, _InputIterator __last, size_type __n, const hasher &__hf, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_map (_InputIterator __first, _InputIterator __last, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- unordered_map (const allocator_type &__a)
- unordered_map (const unordered_map &)=default
- **unordered_map** (const unordered_map &__umap, const allocator_type &__a)
- **unordered_map** (initializer_list< value_type > __l, const allocator_type &__a)
- **unordered_map** (initializer_list< value_type > __l, size_type __n, const allocator_type &__a)
- **unordered_map** (initializer_list< value_type > __l, size_type __n, const hasher &__hf, const allocator_type &__a)
- unordered_map (initializer_list< value_type > __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_map** (size_type __n, const allocator_type &__a)
- **unordered_map** (size_type __n, const hasher &__hf, const allocator_type &__a)
- unordered_map (size_type __n, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- unordered_map (unordered_map &&)=default
- **unordered_map** (unordered_map &&__umap, const allocator_type &__a) noexcept(noexcept(_Hashtable(std::move(__umap._M_h), __a)))
- iterator begin () noexcept
- local_iterator begin (size_type __n)
- size_type **bucket** (const key_type &__key) const
- size_type bucket_count () const noexcept
- size_type **bucket_size** (size_type __n) const
- void clear () noexcept
- template<typename... _Args>
  std::pair< iterator, bool > emplace (_Args &&... __args)
- template<typename... _Args>
  iterator emplace_hint (const_iterator __pos, _Args &&... __args)
- bool empty () const noexcept
- iterator end () noexcept
- local_iterator end (size_type __n)
- size_type erase (const key_type &__x)
- iterator erase (const_iterator __first, const_iterator __last)
- allocator_type get_allocator () const noexcept
- hasher hash_function () const
- template<typename _InputIterator>
  void insert (_InputIterator __first, _InputIterator __last)
- void insert (initializer_list< value_type > __l)
- key_equal key_eq () const
- float load_factor () const noexcept
- size_type max_bucket_count () const noexcept

- float max_load_factor () const noexcept
- void max_load_factor (float __z)
- size_type max_size () const noexcept
- unordered_map & operator= (const unordered_map &)=default
- unordered_map & operator= (initializer_list< value_type > __l)
- unordered_map & operator= (unordered_map &&)=default
- void rehash (size_type __n)
- void reserve (size_type __n)
- size_type size () const noexcept
- void swap (unordered_map &__x) noexcept(noexcept(_M_h.swap(__x._M_h)))

- const_iterator begin () const noexcept
- const_iterator cbegin () const noexcept

- const_iterator end () const noexcept
- const_iterator cend () const noexcept

- std::pair< iterator, bool > insert (const value_type &__x)
- std::pair< iterator, bool > insert (value_type &&__x)
- template<typename _Pair>
  __enable_if_t< is_constructible< value_type, _Pair && >::value, pair< iterator, bool > > insert (_Pair &&__x)

- iterator insert (const_iterator __hint, const value_type &__x)
- iterator insert (const_iterator __hint, value_type &&__x)
- template<typename _Pair>
  __enable_if_t< is_constructible< value_type, _Pair && >::value, iterator > insert (const_iterator __hint, _Pair &&__x)

- iterator erase (const_iterator __position)
- iterator erase (iterator __position)

- iterator find (const key_type &__x)
- template<typename _Kt>
  auto find (const _Kt &__x) -> decltype(_M_h._M_find_tr(__x))
- const_iterator find (const key_type &__x) const
- template<typename _Kt>
  auto find (const _Kt &__x) const -> decltype(_M_h._M_find_tr(__x))

- size_type count (const key_type &__x) const
- template<typename _Kt>
  auto count (const _Kt &__x) const -> decltype(_M_h._M_count_tr(__x))

- bool contains (const key_type &__x) const
- template<typename _Kt>
  auto contains (const _Kt &__x) const -> decltype(_M_h._M_find_tr(__x), void(), true)

- std::pair< iterator, iterator > equal_range (const key_type &__x)
- template<typename _Kt>
  auto equal_range (const _Kt &__x) -> decltype(_M_h._M_equal_range_tr(__x))
- std::pair< const_iterator, const_iterator > equal_range (const key_type &__x) const
- template<typename _Kt>
  auto equal_range (const _Kt &__x) const -> decltype(_M_h._M_equal_range_tr(__x))

- mapped_type & operator[ ] (const key_type &__k)
- mapped_type & operator[ ] (key_type &&__k)

- mapped_type & at (const key_type &__k)
- const mapped_type & at (const key_type &__k) const

- const_local_iterator begin (size_type __n) const
- const_local_iterator cbegin (size_type __n) const

- const_local_iterator end (size_type __n) const
- const_local_iterator cend (size_type __n) const

Friends

- template<typename _Key1, typename _Tp1, typename _Hash1, typename _Pred1, typename _Alloc1>
  bool **operator==** (const unordered_map< _Key1, _Tp1, _Hash1, _Pred1, _Alloc1 > &, const unordered_map< _Key1, _Tp1, _Hash1, _Pred1, _Alloc1 > &)

5.996.1 Detailed Description

template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
class std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >

A standard container composed of unique keys (containing at most one of each key value) that associates values of another type with the keys.

Since

C++11

Template Parameters

_Key	Type of key objects.
_Tp	Type of mapped objects.
_Hash	Hashing function object type, defaults to hash<_Key>.

_Pred	Predicate function object type, defaults to equal_to<_Key>.
_Alloc	Allocator type, defaults to std::allocator<std::pair<const _Key, _Tp>>.

Meets the requirements of a container, and unordered associative container
The resulting value type of the container is std::pair<const _Key, _Tp>.
Base is _Hashtable, dispatched at compile time via template alias __umap_hashtable.

5.996.2 Member Typedef Documentation

allocator_type

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::allocator_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::allocator↩
_type
```

Public typedefs.

const_iterator

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::const_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::const↩
_iterator
```

Iterator-related typedefs.

const_local_iterator

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::const_local_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >↩
::const_local_iterator
```

Iterator-related typedefs.

const_pointer

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::const_pointer std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::const_↩
pointer
```

Iterator-related typedefs.

const_reference

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::const_reference std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::const↩
_reference
```

Iterator-related typedefs.

difference_type

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
```

```
typedef _Hashtable::difference_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::difference↩
_type
```
Iterator-related typedefs.

hasher

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::hasher std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::hasher
```
Public typedefs.

iterator

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::iterator
```
Iterator-related typedefs.

key_equal

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::key_equal std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::key_equal
```
Public typedefs.

key_type

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::key_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::key_type
```
Public typedefs.

local_iterator

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::local_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::local↩
_iterator
```
Iterator-related typedefs.

mapped_type

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::mapped_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::mapped_↩
type
```
Public typedefs.

pointer

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::pointer std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::pointer
```
Iterator-related typedefs.

reference

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::reference std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::reference
```

Iterator-related typedefs.

size_type

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::size_type
```

Iterator-related typedefs.

value_type

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::value_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::value_type
```

Public typedefs.

5.996.3 Constructor & Destructor Documentation

unordered_map() [1/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map ()  [default]
```

Default constructor.

unordered_map() [2/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map (
            size_type __n,
            const hasher & __hf = hasher(),
            const key_equal & __eql = key_equal(),
            const allocator_type & __a = allocator_type())  [inline], [explicit]
```

Default constructor creates no elements.

Parameters

__n	Minimal initial number of buckets.
__hf	A hash functor.
__eql	A key equality functor.
__a	An allocator object.

unordered_map() [3/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _InputIterator>
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map (
```

```
            _InputIterator __first,
            _InputIterator __last,
            size_type __n = 0,
            const hasher & __hf = hasher(),
            const key_equal & __eql = key_equal(),
            const allocator_type & __a = allocator_type())  [inline]
```

Builds an unordered_map from a range.

Parameters

__first	An input iterator.
__last	An input iterator.
__n	Minimal initial number of buckets.
__hf	A hash functor.
__eql	A key equality functor.
__a	An allocator object.

Create an unordered_map consisting of copies of the elements from [__first,__last). This is linear in N (where N is distance(__first,__last)).

unordered_map() [4/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map (
            const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > & )  [default]
```

Copy constructor.

unordered_map() [5/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map (
            unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > && )  [default]
```

Move constructor.

unordered_map() [6/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map (
            const allocator_type & __a)  [inline], [explicit]
```

Creates an unordered_map with no elements.

Parameters

_↩ _a	An allocator object.

unordered_map() [7/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
```

```
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map (
            initializer_list< value_type > __l,
            size_type __n = 0,
            const hasher & __hf = hasher(),
            const key_equal & __eql = key_equal(),
            const allocator_type & __a = allocator_type())  [inline]
```

Builds an unordered_map from an initializer_list.

Parameters

__l	An initializer_list.
__n	Minimal initial number of buckets.
__hf	A hash functor.
__eql	A key equality functor.
__a	An allocator object.

Create an unordered_map consisting of copies of the elements in the list. This is linear in N (where N is *__l.size()*).

5.996.4 Member Function Documentation

at() [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
mapped_type & std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::at (
            const key_type & __k)  [inline]
```

Access to unordered_map data.

Parameters

_↩ _k	The key for which data should be retrieved.

Returns

A reference to the data whose key is equal to *__k*, if such a data is present in the unordered_map.

Exceptions

std::out_of_range	If no such data is present.

at() [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const mapped_type & std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::at (
            const key_type & __k) const  [inline]
```

Access to unordered_map data.

Parameters

_↩ _k	The key for which data should be retrieved.

Returns

A reference to the data whose key is equal to __k, if such a data is present in the unordered_map.

Exceptions

std::out_of_range	If no such data is present.

begin() [1/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::begin () const  [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered_map.

begin() [2/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::begin ()  [inline], [noexcept]
```

Returns a read/write iterator that points to the first element in the unordered_map.

begin() [3/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
local_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::begin (
            size_type __n)  [inline]
```

Returns a read/write iterator pointing to the first bucket element.

Parameters

_↩ _n	The bucket index.

Returns

A read/write local iterator.

begin() [4/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_local_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::begin (
            size_type __n) const  [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

_↩ _n	The bucket index.

Returns

A read-only local iterator.

bucket_count()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::bucket_count () const  [inline],
[noexcept]
```

Returns the number of buckets of the unordered_map.

cbegin() [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::cbegin () const  [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered_map.

cbegin() [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_local_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::cbegin (
            size_type __n) const  [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

_↩ _n	The bucket index.

Returns

A read-only local iterator.

cend() [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::cend () const  [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered_map.

cend() [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_local_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::cend (
            size_type __n) const  [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

_↩ _n	The bucket index.

Returns

A read-only local iterator.

clear()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::clear ()  [inline], [noexcept]
```

Erases all elements in an unordered_map. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

contains() [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _Kt>
auto std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::contains (
            const _Kt & __x) const -> decltype(_M_h._M_find_tr(__x), void(), true)   [inline]
```

Finds whether an element with the given key exists.

Parameters

_↩ _x	Key of elements to be located.

Returns

True if there is any element with the specified key.

contains() [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
bool std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::contains (
            const key_type & __x) const  [inline]
```

Finds whether an element with the given key exists.

Parameters

_↩ _x	Key of elements to be located.

Returns

True if there is any element with the specified key.

count() [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _Kt>
auto std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::count (
            const _Kt & __x) const -> decltype(_M_h._M_count_tr(__x))   [inline]
```

Finds the number of elements.

Parameters

_↩ _x	Key to count.

Returns

Number of elements with specified key.

This function only makes sense for unordered_multimap; for unordered_map the result will either be 0 (not present) or 1 (present).

count() [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::count (
            const key_type & __x) const  [inline]
```

Finds the number of elements.

Parameters

_↩ _x	Key to count.

Returns

Number of elements with specified key.

This function only makes sense for unordered_multimap; for unordered_map the result will either be 0 (not present) or 1 (present).

emplace()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename...  _Args>
std::pair< iterator, bool > std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::emplace (
            _Args &&...  __args)  [inline]
```

Attempts to build and insert a std::pair into the unordered_map.

Parameters

__args	Arguments used to generate a new pair instance (see std::piecewise_contruct for passing arguments to each part of the pair constructor).

Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to build and insert a (key, value) pair into the unordered_map. An unordered_map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the unordered_map.
Insertion requires amortized constant time.

emplace_hint()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename... _Args>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::emplace_hint (
            const_iterator __pos,
            _Args &&... __args)  [inline]
```

Attempts to build and insert a std::pair into the unordered_map.

Parameters

__pos	An iterator that serves as a hint as to where the pair should be inserted.
__args	Arguments used to generate a new pair instance (see std::piecewise_contruct for passing arguments to each part of the pair constructor).

Returns

An iterator that points to the element with key of the std::pair built from *__args* (may or may not be that std::pair).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument emplace() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
See https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.↩associative.insert_hints for more on *hinting*.
Insertion requires amortized constant time.

empty()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
bool std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::empty () const  [inline], [nodiscard],
[noexcept]
```

Returns true if the unordered_map is empty.

end() [1/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::end () const  [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered_map.

end() [2/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::end ()  [inline], [noexcept]
```

Returns a read/write iterator that points one past the last element in the unordered_map.

end() [3/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
local_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::end (
            size_type __n)  [inline]
```

Returns a read/write iterator pointing to one past the last bucket elements.

Parameters

_↩ _n	The bucket index.

Returns

A read/write local iterator.

end() [4/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_local_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::end (
            size_type __n) const  [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

_↩ _n	The bucket index.

Returns

A read-only local iterator.

equal_range() [1/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _Kt>
auto std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::equal_range (
            const _Kt & __x) -> decltype(_M_h._M_equal_range_tr(__x))   [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for unordered_multimap.

equal_range() [2/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _Kt>
auto std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::equal_range (
            const _Kt & __x) const -> decltype(_M_h._M_equal_range_tr(__x))   [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for unordered_multimap.

equal_range() [3/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::pair< iterator, iterator > std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::equal_↩
range (
            const key_type & __x)  [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for unordered_multimap.

equal_range() [4/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::pair< const_iterator, const_iterator > std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc
>::equal_range (
            const key_type & __x) const  [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for unordered_multimap.

erase() [1/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
            const key_type & __x)  [inline]
```

Erases elements according to the provided key.

Parameters

_↩ _x	Key of element to be erased.

Returns

The number of elements erased.

This function erases all the elements located by the given key from an unordered_map. For an unordered_map the result of this function can only be 0 (not present) or 1 (present). Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [2/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
            const_iterator __first,
            const_iterator __last)  [inline]
```

Erases a [__first,__last) range of elements from an unordered_map.

Parameters

__first	Iterator pointing to the start of the range to be erased.
__last	Iterator pointing to the end of the range to be erased.

Returns

The iterator *__last*.

This function erases a sequence of elements from an unordered_map. Note that this function only erases the elements, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [3/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
            const_iterator __position)  [inline]
```

Erases an element from an unordered_map.

Parameters

__position	An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *__position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from an unordered_map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [4/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
            iterator __position)  [inline]
```

Erases an element from an unordered_map.

Parameters

__position	An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *__position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from an unordered_map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

find() [1/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _Kt>
auto std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::find (
            const _Kt & __x) -> decltype(_M_h._M_find_tr(__x))   [inline]
```

Tries to locate an element in an unordered_map.

Parameters

_↩ _x	Key to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

find() [2/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _Kt>
auto std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::find (
            const _Kt & __x) const -> decltype(_M_h._M_find_tr(__x))   [inline]
```

Tries to locate an element in an unordered_map.

Parameters

_↩ _x	Key to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

find() [3/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::find (
            const key_type & __x)  [inline]
```

Tries to locate an element in an unordered_map.

Parameters

_↩ _x	Key to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

find() [4/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::find (
            const key_type & __x) const  [inline]
```

Tries to locate an element in an unordered_map.

Parameters

_↩ _x	Key to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

get_allocator()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
allocator_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::get_allocator () const
[inline], [noexcept]
```

Returns the allocator object used by the unordered_map.

hash_function()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
hasher std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::hash_function () const  [inline]
```

Returns the hash functor object with which the unordered_map was constructed.

insert() [1/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _InputIterator>
void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
            _InputIterator __first,
            _InputIterator __last)  [inline]
```

A template function that attempts to insert a range of elements.

Parameters

__first	Iterator pointing to the start of the range to be inserted.
__last	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

insert() [2/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _Pair>
__enable_if_t< is_constructible< value_type, _Pair && >::value, pair< iterator, bool > > std::unordered_map<
_Key, _Tp, _Hash, _Pred, _Alloc >::insert (
            _Pair && __x)  [inline]
```

Attempts to insert a std::pair into the unordered_map.

Parameters

_↩ _x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the unordered_map. An unordered_map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the unordered_map.
Insertion requires amortized constant time.

insert() [3/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::pair< iterator, bool > std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
            const value_type & __x)  [inline]
```

Attempts to insert a std::pair into the unordered_map.

Parameters

_↩ _x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the unordered_map. An unordered_map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the unordered_map.
Insertion requires amortized constant time.

insert() [4/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _Pair>
__enable_if_t< is_constructible< value_type, _Pair && >::value, iterator > std::unordered_map<
_Key, _Tp, _Hash, _Pred, _Alloc >::insert (
            const_iterator __hint,
            _Pair && __x)  [inline]
```

Attempts to insert a std::pair into the unordered_map.

Parameters

__hint	An iterator that serves as a hint as to where the pair should be inserted.
__x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the element with key of __x (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument insert() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
See https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints for more on *hinting*.
Insertion requires amortized constant time.

insert() [5/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
            const_iterator __hint,
            const value_type & __x)  [inline]
```

Attempts to insert a std::pair into the unordered_map.

Parameters

__hint	An iterator that serves as a hint as to where the pair should be inserted.
__x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the element with key of __x (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument insert() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
See https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints for more on *hinting*.
Insertion requires amortized constant time.

insert() [6/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
            const_iterator __hint,
            value_type && __x)  [inline]
```

Attempts to insert a std::pair into the unordered_map.

Parameters

__hint	An iterator that serves as a hint as to where the pair should be inserted.
__x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the element with key of __x (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument insert() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
See https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints for more on *hinting*.
Insertion requires amortized constant time.

insert() [7/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
            initializer_list< value_type > __l)  [inline]
```

Attempts to insert a list of elements into the unordered_map.

Parameters

__l	A std::initializer_list<value_type> of elements to be inserted.

Complexity similar to that of the range constructor.

insert() [8/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::pair< iterator, bool > std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
            value_type && __x)  [inline]
```

Attempts to insert a std::pair into the unordered_map.

Parameters

__x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the unordered_map. An unordered_map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the unordered_map.
Insertion requires amortized constant time.

key_eq()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
key_equal std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::key_eq () const  [inline]
```

Returns the key comparison object with which the unordered_map was constructed.

load_factor()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
float std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::load_factor () const  [inline],
[noexcept]
```

Returns the average number of elements per bucket.

max_bucket_count()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::max_bucket_count () const  [inline],
[noexcept]
```

Returns the maximum number of buckets of the unordered_map.

max_load_factor() [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
float std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::max_load_factor () const  [inline],
[noexcept]
```

Returns a positive number that the unordered_map tries to keep the load factor less than or equal to.

max_load_factor() [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::max_load_factor (
            float __z)  [inline]
```

Change the unordered_map maximum load factor.

Parameters

_↩ _z	The new maximum load factor.

max_size()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::max_size () const  [inline],
[noexcept]
```

Returns the maximum size of the unordered_map.

operator=() [1/3]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
unordered_map & std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::operator= (
            const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > & )  [default]
```

Copy assignment operator.

operator=() [2/3]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
unordered_map & std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::operator= (
            initializer_list< value_type > __l)  [inline]
```

Unordered_map list assignment operator.

Parameters

↩ _↩ ↩ _↩ *l*	An initializer_list.

This function fills an unordered_map with copies of the elements in the initializer list *__l*.
Note that the assignment completely changes the unordered_map and that the resulting unordered_map's size is the same as the number of elements assigned.

operator=() [3/3]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
unordered_map & std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::operator= (
            unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > && )  [default]
```

Move assignment operator.

operator[]() [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
mapped_type & std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::operator[] (
            const key_type & __k)  [inline]
```

Subscript ([]) access to unordered_map data.

Parameters

_↩ *_k*	The key for which data should be retrieved.

Returns

A reference to the data of the (key,data) pair.

Allows for easy lookup with the subscript ([])operator. Returns data associated with the key specified in subscript. If the key does not exist, a pair with that key is created using default values, which is then returned.
Lookup requires constant time.

operator[]() [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
mapped_type & std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::operator[] (
            key_type && __k)  [inline]
```

Subscript ([]) access to unordered_map data.

Parameters

_↩ _k	The key for which data should be retrieved.

Returns

A reference to the data of the (key,data) pair.

Allows for easy lookup with the subscript ([])operator. Returns data associated with the key specified in subscript. If the key does not exist, a pair with that key is created using default values, which is then returned.
Lookup requires constant time.

rehash()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::rehash (
            size_type __n)  [inline]
```

May rehash the unordered_map.

Parameters

_↩ _n	The new number of buckets.

Rehash will occur only if the new number of buckets respect the unordered_map maximum load factor.

reserve()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::reserve (
            size_type __n)  [inline]
```

Prepare the unordered_map for a specified number of elements.

Parameters

_↩ _n	Number of elements required.

Same as rehash(ceil(n / max_load_factor())).

size()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::size () const  [inline], [noexcept]
```

Returns the size of the unordered_map.

swap()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
```

```
void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::swap (
            unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > & __x)  [inline], [noexcept]
```

Swaps data with another unordered_map.

Parameters

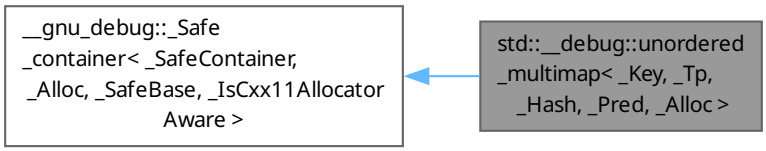
_↩ _x	An unordered_map of the same element and allocator types.

This exchanges the elements between two unordered_map in constant time. Note that the global std::swap() function is specialized such that std::swap(m1,m2) will feed to this function.
The documentation for this class was generated from the following file:

- unordered_map.h

5.997 std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference

`#include <unordered_map>`
Inheritance diagram for std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >:



Public Types

- typedef _Base::allocator_type **allocator_type**
- typedef __gnu_debug::_Safe_iterator< _Base_const_iterator, unordered_multimap > **const_iterator**
- typedef __gnu_debug::_Safe_local_iterator< _Base_const_local_iterator, unordered_multimap > **const_local_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**
- typedef _Base::difference_type **difference_type**
- typedef _Base::hasher **hasher**
- typedef __gnu_debug::_Safe_iterator< _Base_iterator, unordered_multimap > **iterator**
- typedef _Base::key_equal **key_equal**
- typedef _Base::key_type **key_type**
- typedef __gnu_debug::_Safe_local_iterator< _Base_local_iterator, unordered_multimap > **local_iterator**
- typedef _Base::mapped_type **mapped_type**
- using **node_type**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef _Base::size_type **size_type**
- typedef _Base::value_type **value_type**

Public Member Functions

- **unordered_multimap** (_Base_ref __x)
- template<typename _InputIterator>
  unordered_multimap (_InputIterator __first, _InputIterator __last, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_multimap (_InputIterator __first, _InputIterator __last, size_type __n, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_multimap (_InputIterator __first, _InputIterator __last, size_type __n, const hasher &__hf, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_multimap (_InputIterator __first, _InputIterator __last, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_multimap** (const allocator_type &__a)
- **unordered_multimap** (const unordered_multimap &)=default
- **unordered_multimap** (const unordered_multimap &__umap, const allocator_type &__a)
- **unordered_multimap** (initializer_list< value_type > __l, const allocator_type &__a)
- **unordered_multimap** (initializer_list< value_type > __l, size_type __n, const allocator_type &__a)
- **unordered_multimap** (initializer_list< value_type > __l, size_type __n, const hasher &__hf, const allocator_type &__a)
- **unordered_multimap** (initializer_list< value_type > __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_multimap** (size_type __n, const allocator_type &__a)
- **unordered_multimap** (size_type __n, const hasher &__hf, const allocator_type &__a)
- **unordered_multimap** (size_type __n, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_multimap** (unordered_multimap &&)=default
- **unordered_multimap** (unordered_multimap &&__umap, const allocator_type &__a) noexcept(noexcept(_Base(std::move(__umap), __a)))
- const _Base & **_M_base** () const noexcept
- _Base & **_M_base** () noexcept
- const_iterator **begin** () const noexcept
- iterator **begin** () noexcept
- local_iterator **begin** (size_type __b)
- const_local_iterator **begin** (size_type __b) const
- size_type **bucket_size** (size_type __b) const
- const_iterator **cbegin** () const noexcept
- const_local_iterator **cbegin** (size_type __b) const
- const_iterator **cend** () const noexcept
- const_local_iterator **cend** (size_type __b) const
- void **clear** () noexcept
- template<typename... _Args>
  iterator **emplace** (_Args &&... __args)
- template<typename... _Args>
  iterator **emplace_hint** (const_iterator __hint, _Args &&... __args)
- const_iterator **end** () const noexcept
- iterator **end** () noexcept
- local_iterator **end** (size_type __b)
- const_local_iterator **end** (size_type __b) const
- template<typename _Kt, typename = std::__has_is_transparent_t<_Hash, _Kt>, typename = std::__has_is_transparent_t<_Pred, _Kt>>
  std::pair< iterator, iterator > **equal_range** (const _Kt &__k)

- template<typename _Kt, typename = std::__has_is_transparent_t<_Hash, _Kt>, typename = std::__has_is_transparent_t<_Pred, _Kt>>
  std::pair< const_iterator, const_iterator > **equal_range** (const _Kt &__k) const
- std::pair< iterator, iterator > **equal_range** (const key_type &__key)
- std::pair< const_iterator, const_iterator > **equal_range** (const key_type &__key) const
- _Base_iterator **erase** (_Base_const_iterator __it)
- size_type **erase** (const key_type &__key)
- iterator **erase** (const_iterator __first, const_iterator __last)
- iterator **erase** (const_iterator __it)
- iterator **erase** (iterator __it)
- node_type **extract** (const key_type &__key)
- node_type **extract** (const_iterator __position)
- template<typename _Kt, typename = std::__has_is_transparent_t<_Hash, _Kt>, typename = std::__has_is_transparent_t<_Pred, _Kt>>
  iterator **find** (const _Kt &__k)
- template<typename _Kt, typename = std::__has_is_transparent_t<_Hash, _Kt>, typename = std::__has_is_transparent_t<_Pred, _Kt>>
  const_iterator **find** (const _Kt &__k) const
- iterator **find** (const key_type &__key)
- const_iterator **find** (const key_type &__key) const
- template<typename _InputIterator>
  void **insert** (_InputIterator __first, _InputIterator __last)
- template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
  iterator **insert** (_Pair &&__obj)
- iterator **insert** (const value_type &__obj)
- template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
  iterator **insert** (const_iterator __hint, _Pair &&__obj)
- iterator **insert** (const_iterator __hint, const value_type &__obj)
- iterator **insert** (const_iterator __hint, node_type &&__nh)
- iterator **insert** (const_iterator __hint, value_type &&__x)
- iterator **insert** (node_type &&__nh)
- void **insert** (std::initializer_list< value_type > __l)
- iterator **insert** (value_type &&__x)
- float **max_load_factor** () const noexcept
- void **max_load_factor** (float __f)
- template<typename _H2, typename _P2>
  void **merge** (unordered_map< _Key, _Tp, _H2, _P2, _Alloc > &&__source)
- template<typename _H2, typename _P2>
  void **merge** (unordered_map< _Key, _Tp, _H2, _P2, _Alloc > &__source)
- template<typename _H2, typename _P2>
  void **merge** (unordered_multimap< _Key, _Tp, _H2, _P2, _Alloc > &&__source)
- template<typename _H2, typename _P2>
  void **merge** (unordered_multimap< _Key, _Tp, _H2, _P2, _Alloc > &__source)
- unordered_multimap & **operator=** (const unordered_multimap &)=default
- unordered_multimap & **operator=** (initializer_list< value_type > __l)
- unordered_multimap & **operator=** (unordered_multimap &&)=default
- void **swap** (unordered_multimap &__x) noexcept(noexcept(declval< _Base & >().swap(__x)))

Protected Member Functions

- constexpr void **_M_swap** (const _Safe_container &__x) const noexcept

Friends

- template<typename _ItT, typename _SeqT, typename _CatT>
  class **::__gnu_debug::_Safe_iterator**
- template<typename _ItT, typename _SeqT>
  class **::__gnu_debug::_Safe_local_iterator**

5.997.1 Detailed Description

template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >>
class std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >

Class std::unordered_multimap with safety/checking/debug instrumentation.
The documentation for this class was generated from the following file:

- debug/unordered_map

5.998 std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference

`#include <unordered_map>`

Public Types

- typedef _Hashtable::key_type key_type
- typedef _Hashtable::value_type value_type
- typedef _Hashtable::mapped_type mapped_type
- typedef _Hashtable::hasher hasher
- typedef _Hashtable::key_equal key_equal
- typedef _Hashtable::allocator_type allocator_type

- typedef _Hashtable::pointer pointer
- typedef _Hashtable::const_pointer const_pointer
- typedef _Hashtable::reference reference
- typedef _Hashtable::const_reference const_reference
- typedef _Hashtable::iterator iterator
- typedef _Hashtable::const_iterator const_iterator
- typedef _Hashtable::local_iterator local_iterator
- typedef _Hashtable::const_local_iterator const_local_iterator
- typedef _Hashtable::size_type size_type
- typedef _Hashtable::difference_type difference_type

Public Member Functions

- unordered_multimap ()=default
- template<typename _InputIterator>
  unordered_multimap (_InputIterator __first, _InputIterator __last, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_multimap (_InputIterator __first, _InputIterator __last, size_type __n, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_multimap (_InputIterator __first, _InputIterator __last, size_type __n, const hasher &__hf, const allocator_type &__a)

- template<typename _InputIterator>
  unordered_multimap (_InputIterator __first, _InputIterator __last, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- unordered_multimap (const allocator_type &__a)
- unordered_multimap (const unordered_multimap &)=default
- **unordered_multimap** (const unordered_multimap &__ummap, const allocator_type &__a)
- **unordered_multimap** (initializer_list< value_type > __l, const allocator_type &__a)
- **unordered_multimap** (initializer_list< value_type > __l, size_type __n, const allocator_type &__a)
- **unordered_multimap** (initializer_list< value_type > __l, size_type __n, const hasher &__hf, const allocator_type &__a)
- unordered_multimap (initializer_list< value_type > __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_multimap** (size_type __n, const allocator_type &__a)
- **unordered_multimap** (size_type __n, const hasher &__hf, const allocator_type &__a)
- unordered_multimap (size_type __n, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- unordered_multimap (unordered_multimap &&)=default
- **unordered_multimap** (unordered_multimap &&__ummap, const allocator_type &__a) noexcept(noexcept(_↩ Hashtable(std::move(__ummap._M_h), __a)))
- iterator begin () noexcept
- local_iterator begin (size_type __n)
- size_type **bucket** (const key_type &__key) const
- size_type bucket_count () const noexcept
- size_type **bucket_size** (size_type __n) const
- void clear () noexcept
- template<typename... _Args>
  iterator emplace (_Args &&... __args)
- template<typename... _Args>
  iterator emplace_hint (const_iterator __pos, _Args &&... __args)
- bool empty () const noexcept
- iterator end () noexcept
- local_iterator end (size_type __n)
- size_type erase (const key_type &__x)
- iterator erase (const_iterator __first, const_iterator __last)
- allocator_type get_allocator () const noexcept
- hasher hash_function () const
- template<typename _InputIterator>
  void insert (_InputIterator __first, _InputIterator __last)
- void insert (initializer_list< value_type > __l)
- key_equal key_eq () const
- float load_factor () const noexcept
- size_type max_bucket_count () const noexcept
- float max_load_factor () const noexcept
- void max_load_factor (float __z)
- size_type max_size () const noexcept
- unordered_multimap & operator= (const unordered_multimap &)=default
- unordered_multimap & operator= (initializer_list< value_type > __l)
- unordered_multimap & operator= (unordered_multimap &&)=default
- void rehash (size_type __n)
- void reserve (size_type __n)
- size_type size () const noexcept

- void swap (unordered_multimap &__x) noexcept(noexcept(_M_h.swap(__x._M_h)))

- const_iterator begin () const noexcept
- const_iterator cbegin () const noexcept

- const_iterator end () const noexcept
- const_iterator cend () const noexcept

- iterator insert (const value_type &__x)
- iterator insert (value_type &&__x)
- template<typename _Pair>
  __enable_if_t< is_constructible< value_type, _Pair && >::value, iterator > insert (_Pair &&__x)

- iterator insert (const_iterator __hint, const value_type &__x)
- iterator insert (const_iterator __hint, value_type &&__x)
- template<typename _Pair>
  __enable_if_t< is_constructible< value_type, _Pair && >::value, iterator > insert (const_iterator __hint, _Pair &&__x)

- iterator erase (const_iterator __position)
- iterator erase (iterator __position)

- iterator find (const key_type &__x)
- template<typename _Kt>
  auto find (const _Kt &__x) -> decltype(_M_h._M_find_tr(__x))
- const_iterator find (const key_type &__x) const
- template<typename _Kt>
  auto find (const _Kt &__x) const -> decltype(_M_h._M_find_tr(__x))

- size_type count (const key_type &__x) const
- template<typename _Kt>
  auto count (const _Kt &__x) const -> decltype(_M_h._M_count_tr(__x))

- bool contains (const key_type &__x) const
- template<typename _Kt>
  auto contains (const _Kt &__x) const -> decltype(_M_h._M_find_tr(__x), void(), true)

- std::pair< iterator, iterator > equal_range (const key_type &__x)
- template<typename _Kt>
  auto equal_range (const _Kt &__x) -> decltype(_M_h._M_equal_range_tr(__x))

- std::pair< const_iterator, const_iterator > equal_range (const key_type &__x) const
- template<typename _Kt>
  auto equal_range (const _Kt &__x) const -> decltype(_M_h._M_equal_range_tr(__x))

- const_local_iterator begin (size_type __n) const
- const_local_iterator cbegin (size_type __n) const

- const_local_iterator end (size_type __n) const
- const_local_iterator cend (size_type __n) const

Friends

- template<typename _Key1, typename _Tp1, typename _Hash1, typename _Pred1, typename _Alloc1>
  bool **operator==** (const unordered_multimap< _Key1, _Tp1, _Hash1, _Pred1, _Alloc1 > &, const unordered_multimap< _Key1, _Tp1, _Hash1, _Pred1, _Alloc1 > &)

5.998.1 Detailed Description

template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
class std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >

A standard container composed of equivalent keys (possibly containing multiple of each key value) that associates values of another type with the keys.

Since

C++11

Template Parameters

_Key	Type of key objects.
_Tp	Type of mapped objects.
_Hash	Hashing function object type, defaults to hash<_Key>.
_Pred	Predicate function object type, defaults to equal_to<_Key>.
_Alloc	Allocator type, defaults to std::allocator<std::pair<const _Key, _Tp>>.

Meets the requirements of a `container`, and `unordered associative container`
The resulting value type of the container is std::pair<const _Key, _Tp>.
Base is _Hashtable, dispatched at compile time via template alias __ummap_hashtable.

5.998.2 Member Typedef Documentation

allocator_type

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::allocator_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::allocator_type
```

Public typedefs.

const_iterator

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::const_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >↩
::const_iterator
```

Iterator-related typedefs.

const_local_iterator

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::const_local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc
>::const_local_iterator
```

Iterator-related typedefs.

const_pointer

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::const_pointer std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >↩
::const_pointer
```

Iterator-related typedefs.

const_reference

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::const_reference std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >↩
::const_reference
```

Iterator-related typedefs.

difference_type

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::difference_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >↩
::difference_type
```

Iterator-related typedefs.

hasher

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::hasher std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::hasher
```

Public typedefs.

iterator

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::iterator
```

Iterator-related typedefs.

key_equal

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::key_equal std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::key_↩
equal
```

Public typedefs.

key_type

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::key_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::key_type
```

Public typedefs.

local_iterator

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >↩
::local_iterator
```

Iterator-related typedefs.

mapped_type

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::mapped_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >↩
::mapped_type
```

Public typedefs.

pointer

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::pointer std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::pointer
```

Iterator-related typedefs.

reference

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::reference std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::reference
```

Iterator-related typedefs.

size_type

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::size_↩
type
```

Iterator-related typedefs.

value_type

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::value_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::value↩
_type
```

Public typedefs.

5.998.3 Constructor & Destructor Documentation

unordered_multimap() [1/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap ()  [default]
```

Default constructor.

unordered_multimap() [2/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap (
            size_type __n,
            const hasher & __hf = hasher(),
            const key_equal & __eql = key_equal(),
            const allocator_type & __a = allocator_type())  [inline], [explicit]
```

Default constructor creates no elements.

Parameters

__n	Mnimal initial number of buckets.
__hf	A hash functor.
__eql	A key equality functor.
__a	An allocator object.

unordered_multimap() [3/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _InputIterator>
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap (
            _InputIterator __first,
            _InputIterator __last,
            size_type __n = 0,
            const hasher & __hf = hasher(),
            const key_equal & __eql = key_equal(),
            const allocator_type & __a = allocator_type())  [inline]
```

Builds an unordered_multimap from a range.

Parameters

__first	An input iterator.
__last	An input iterator.
__n	Minimal initial number of buckets.

Parameters

__hf	A hash functor.
__eql	A key equality functor.
__a	An allocator object.

Create an unordered_multimap consisting of copies of the elements from [__first,__last). This is linear in N (where N is distance(__first,__last)).

unordered_multimap() [4/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap (
            const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > & )  [default]
```

Copy constructor.

unordered_multimap() [5/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap (
            unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > && )  [default]
```

Move constructor.

unordered_multimap() [6/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap (
            const allocator_type & __a)  [inline], [explicit]
```

Creates an unordered_multimap with no elements.

Parameters

_↩ _a	An allocator object.

unordered_multimap() [7/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap (
            initializer_list< value_type > __l,
            size_type __n = 0,
            const hasher & __hf = hasher(),
            const key_equal & __eql = key_equal(),
            const allocator_type & __a = allocator_type())  [inline]
```

Builds an unordered_multimap from an initializer_list.

Parameters

__l	An initializer_list.

Parameters

__n	Minimal initial number of buckets.
__hf	A hash functor.
__eql	A key equality functor.
__a	An allocator object.

Create an unordered_multimap consisting of copies of the elements in the list. This is linear in N (where N is *__l.size()*).

5.998.4 Member Function Documentation

begin() [1/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::begin () const  [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered_multimap.

begin() [2/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::begin ()  [inline], [noexcept]
```

Returns a read/write iterator that points to the first element in the unordered_multimap.

begin() [3/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::begin (
            size_type __n)  [inline]
```

Returns a read/write iterator pointing to the first bucket element.

Parameters

_↩ _n	The bucket index.

Returns

A read/write local iterator.

begin() [4/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::begin (
            size_type __n) const  [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

_↩ _n	The bucket index.

Returns

A read-only local iterator.

bucket_count()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::bucket_count () const  [inline],
[noexcept]
```

Returns the number of buckets of the unordered_multimap.

cbegin() [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::cbegin () const  [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered_multimap.

cbegin() [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::cbegin (
            size_type __n) const  [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

_↩ _n	The bucket index.

Returns

A read-only local iterator.

cend() [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::cend () const  [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered_multimap.

cend() [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::cend (
            size_type __n) const  [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

_↩ _n	The bucket index.

Returns

A read-only local iterator.

clear()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::clear ()  [inline], [noexcept]
```

Erases all elements in an unordered_multimap. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

contains() [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _Kt>
auto std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::contains (
            const _Kt & __x) const -> decltype(_M_h._M_find_tr(__x), void(), true)   [inline]
```

Finds whether an element with the given key exists.

Parameters

_↩ _x	Key of elements to be located.

Returns

True if there is any element with the specified key.

contains() [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
bool std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::contains (
            const key_type & __x) const  [inline]
```

Finds whether an element with the given key exists.

Parameters

_↩ _x	Key of elements to be located.

Returns

True if there is any element with the specified key.

count() [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _Kt>
auto std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::count (
            const _Kt & __x) const -> decltype(_M_h._M_count_tr(__x))   [inline]
```

Finds the number of elements.

Parameters

_↩ _x	Key to count.

Returns

Number of elements with specified key.

count() [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::count (
            const key_type & __x) const  [inline]
```

Finds the number of elements.

Parameters

_↩ _x	Key to count.

Returns

Number of elements with specified key.

emplace()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename... _Args>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::emplace (
            _Args &&... __args)  [inline]
```

Attempts to build and insert a std::pair into the unordered_multimap.

Parameters

__args	Arguments used to generate a new pair instance (see std::piecewise_contruct for passing arguments to each part of the pair constructor).

Returns

An iterator that points to the inserted pair.

This function attempts to build and insert a (key, value) pair into the unordered_multimap.
Insertion requires amortized constant time.

emplace_hint()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename... _Args>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::emplace_hint (
            const_iterator __pos,
            _Args &&... __args)  [inline]
```

Attempts to build and insert a std::pair into the unordered_multimap.

Parameters

__pos	An iterator that serves as a hint as to where the pair should be inserted.
__args	Arguments used to generate a new pair instance (see std::piecewise_contruct for passing arguments to each part of the pair constructor).

Returns

An iterator that points to the element with key of the std::pair built from *__args*.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
See https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.↩associative.insert_hints for more on *hinting*.
Insertion requires amortized constant time.

empty()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
bool std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::empty () const  [inline], [nodiscard],
[noexcept]
```

Returns true if the unordered_multimap is empty.

end() [1/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::end () const  [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered_multimap.

end() [2/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::end ()  [inline], [noexcept]
```

Returns a read/write iterator that points one past the last element in the unordered_multimap.

end() [3/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::end (
            size_type __n)  [inline]
```

Returns a read/write iterator pointing to one past the last bucket elements.

Parameters

_↩ _n	The bucket index.

Returns

A read/write local iterator.

end() [4/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::end (
            size_type __n) const  [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

_↩ _n	The bucket index.

Returns

A read-only local iterator.

equal_range() [1/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _Kt>
auto std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::equal_range (
            const _Kt & __x) -> decltype(_M_h._M_equal_range_tr(__x))   [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

equal_range() [2/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _Kt>
auto std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::equal_range (
            const _Kt & __x) const -> decltype(_M_h._M_equal_range_tr(__x))   [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

equal_range() [3/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::pair< iterator, iterator > std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >↩
::equal_range (
            const key_type & __x)  [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

equal_range() [4/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::pair< const_iterator, const_iterator > std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _↩
Alloc >::equal_range (
            const key_type & __x) const  [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

erase() [1/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
            const key_type & __x)  [inline]
```

Erases elements according to the provided key.

Parameters

_↩ _x	Key of elements to be erased.

Returns

The number of elements erased.

This function erases all the elements located by the given key from an unordered_multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [2/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
            const_iterator __first,
            const_iterator __last)  [inline]
```

Erases a [__first,__last) range of elements from an unordered_multimap.

Parameters

__first	Iterator pointing to the start of the range to be erased.
__last	Iterator pointing to the end of the range to be erased.

Returns

The iterator *__last*.

This function erases a sequence of elements from an unordered_multimap. Note that this function only erases the elements, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [3/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
            const_iterator __position)  [inline]
```

Erases an element from an unordered_multimap.

Parameters

__position	An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *__position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from an unordered_multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [4/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
            iterator __position)  [inline]
```

Erases an element from an unordered_multimap.

Parameters

__position	An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *__position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from an unordered_multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

find() [1/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _Kt>
auto std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::find (
             const _Kt & __x) -> decltype(_M_h._M_find_tr(__x))   [inline]
```

Tries to locate an element in an unordered_multimap.

Parameters

_↩ _x	Key to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

find() [2/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _Kt>
auto std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::find (
            const _Kt & __x) const -> decltype(_M_h._M_find_tr(__x))   [inline]
```

Tries to locate an element in an unordered_multimap.

Parameters

_↩ _x	Key to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

find() [3/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::find (
            const key_type & __x)  [inline]
```

Tries to locate an element in an unordered_multimap.

Parameters

_↩ _x	Key to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

find() [4/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::find (
            const key_type & __x) const  [inline]
```

Tries to locate an element in an unordered_multimap.

Parameters

_↩ _x	Key to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

get_allocator()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
allocator_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::get_allocator () const
[inline], [noexcept]
```

Returns the allocator object used by the unordered_multimap.

hash_function()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
hasher std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::hash_function () const  [inline]
```

Returns the hash functor object with which the unordered_multimap was constructed.

insert() [1/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _InputIterator>
void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
            _InputIterator __first,
            _InputIterator __last)  [inline]
```

A template function that attempts to insert a range of elements.

Parameters

__first	Iterator pointing to the start of the range to be inserted.
__last	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

insert() [2/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _Pair>
__enable_if_t< is_constructible< value_type, _Pair && >::value, iterator > std::unordered_multimap<
_Key, _Tp, _Hash, _Pred, _Alloc >::insert (
            _Pair && __x)  [inline]
```

Inserts a std::pair into the unordered_multimap.

Parameters

_↩ _x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the inserted pair.

Insertion requires amortized constant time.

insert() [3/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
            const value_type & __x)  [inline]
```

Inserts a std::pair into the unordered_multimap.

Parameters

_↩ _x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the inserted pair.

Insertion requires amortized constant time.

insert() [4/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _Pair>
__enable_if_t< is_constructible< value_type, _Pair && >::value, iterator > std::unordered_multimap<
_Key, _Tp, _Hash, _Pred, _Alloc >::insert (
            const_iterator __hint,
            _Pair && __x)  [inline]
```

Inserts a std::pair into the unordered_multimap.

Parameters

__hint	An iterator that serves as a hint as to where the pair should be inserted.
__x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the element with key of *__x* (may or may not be the pair passed in).

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
See https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.↩associative.insert_hints for more on *hinting*.
Insertion requires amortized constant time.

insert() [5/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
```

```
            const_iterator __hint,
            const value_type & __x)  [inline]
```

Inserts a std::pair into the unordered_multimap.

Parameters

__hint	An iterator that serves as a hint as to where the pair should be inserted.
__x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the element with key of *__x* (may or may not be the pair passed in).

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
See https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints for more on *hinting*.
Insertion requires amortized constant time.

insert() [6/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
            const_iterator __hint,
            value_type && __x)  [inline]
```

Inserts a std::pair into the unordered_multimap.

Parameters

__hint	An iterator that serves as a hint as to where the pair should be inserted.
__x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the element with key of *__x* (may or may not be the pair passed in).

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
See https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints for more on *hinting*.
Insertion requires amortized constant time.

insert() [7/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
            initializer_list< value_type > __l)  [inline]
```

Attempts to insert a list of elements into the unordered_multimap.

Parameters

__l	A std::initializer_list<value_type> of elements to be inserted.

Complexity similar to that of the range constructor.

insert() [8/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
            value_type && __x)  [inline]
```

Inserts a std::pair into the unordered_multimap.

Parameters

_↩ _x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the inserted pair.

Insertion requires amortized constant time.

key_eq()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
key_equal std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::key_eq () const  [inline]
```

Returns the key comparison object with which the unordered_multimap was constructed.

load_factor()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
float std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::load_factor () const  [inline],
[noexcept]
```

Returns the average number of elements per bucket.

max_bucket_count()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::max_bucket_count () const
[inline], [noexcept]
```

Returns the maximum number of buckets of the unordered_multimap.

max_load_factor() [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
float std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::max_load_factor () const  [inline],
[noexcept]
```

Returns a positive number that the unordered_multimap tries to keep the load factor less than or equal to.

max_load_factor() [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::max_load_factor (
            float __z)  [inline]
```

Change the unordered_multimap maximum load factor.

Parameters

_↩ _z	The new maximum load factor.

max_size()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::max_size () const  [inline],
[noexcept]
```

Returns the maximum size of the unordered_multimap.

operator=() [1/3]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
unordered_multimap & std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::operator= (
            const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > & )  [default]
```

Copy assignment operator.

operator=() [2/3]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
unordered_multimap & std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::operator= (
            initializer_list< value_type > __l)  [inline]
```

Unordered_multimap list assignment operator.

Parameters

↩ _↩ ↩ _↩ *l*	An initializer_list.

This function fills an unordered_multimap with copies of the elements in the initializer list *__l*.
Note that the assignment completely changes the unordered_multimap and that the resulting unordered_multimap's size is the same as the number of elements assigned.

operator=() [3/3]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
unordered_multimap & std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::operator= (
            unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > && )  [default]
```

Move assignment operator.

rehash()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
```

```
void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::rehash (
            size_type __n)  [inline]
```

May rehash the unordered_multimap.

Parameters

_↩ _n	The new number of buckets.

Rehash will occur only if the new number of buckets respect the unordered_multimap maximum load factor.

reserve()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::reserve (
            size_type __n)  [inline]
```

Prepare the unordered_multimap for a specified number of elements.

Parameters

_↩ _n	Number of elements required.

Same as rehash(ceil(n / max_load_factor())).

size()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::size () const  [inline],
[noexcept]
```

Returns the size of the unordered_multimap.

swap()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::swap (
            unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > & __x)  [inline], [noexcept]
```

Swaps data with another unordered_multimap.

Parameters

_↩ _x	An unordered_multimap of the same element and allocator types.

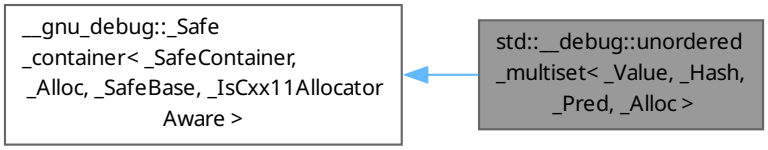
This exchanges the elements between two unordered_multimap in constant time. Note that the global std::swap() function is specialized such that std::swap(m1,m2) will feed to this function.
The documentation for this class was generated from the following file:

- unordered_map.h

5.999 std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc > Class Template Reference

`#include <unordered_set>`

Inheritance diagram for std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >:



Public Types

- typedef _Base::allocator_type **allocator_type**
- typedef __gnu_debug::_Safe_iterator< _Base_const_iterator, unordered_multiset > **const_iterator**
- typedef __gnu_debug::_Safe_local_iterator< _Base_const_local_iterator, unordered_multiset > **const_local_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**
- typedef _Base::difference_type **difference_type**
- typedef _Base::hasher **hasher**
- typedef __gnu_debug::_Safe_iterator< _Base_iterator, unordered_multiset > **iterator**
- typedef _Base::key_equal **key_equal**
- typedef _Base::key_type **key_type**
- typedef __gnu_debug::_Safe_local_iterator< _Base_local_iterator, unordered_multiset > **local_iterator**
- using **node_type**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef _Base::size_type **size_type**
- typedef _Base::value_type **value_type**

Public Member Functions

- **unordered_multiset** (_Base_ref __x)
- template<typename _InputIterator>
  unordered_multiset (_InputIterator __first, _InputIterator __last, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_multiset (_InputIterator __first, _InputIterator __last, size_type __n, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_multiset (_InputIterator __first, _InputIterator __last, size_type __n, const hasher &__hf, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_multiset (_InputIterator __first, _InputIterator __last, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_multiset** (const allocator_type &__a)
- **unordered_multiset** (const unordered_multiset &)=default

- **unordered_multiset** (const unordered_multiset &__uset, const allocator_type &__a)
- **unordered_multiset** (initializer_list< value_type > __l, const allocator_type &__a)
- **unordered_multiset** (initializer_list< value_type > __l, size_type __n, const allocator_type &__a)
- **unordered_multiset** (initializer_list< value_type > __l, size_type __n, const hasher &__hf, const allocator_type &__a)
- **unordered_multiset** (initializer_list< value_type > __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_multiset** (size_type __n, const allocator_type &__a)
- **unordered_multiset** (size_type __n, const hasher &__hf, const allocator_type &__a)
- **unordered_multiset** (size_type __n, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_multiset** (unordered_multiset &&)=default
- **unordered_multiset** (unordered_multiset &&__uset, const allocator_type &__a) noexcept(noexcept(_Base(std::move(↩ __uset), __a)))
- const _Base & **_M_base** () const noexcept
- _Base & **_M_base** () noexcept
- const_iterator **begin** () const noexcept
- iterator **begin** () noexcept
- local_iterator **begin** (size_type __b)
- const_local_iterator **begin** (size_type __b) const
- size_type **bucket_size** (size_type __b) const
- const_iterator **cbegin** () const noexcept
- const_local_iterator **cbegin** (size_type __b) const
- const_iterator **cend** () const noexcept
- const_local_iterator **cend** (size_type __b) const
- void **clear** () noexcept
- template<typename... _Args>
  iterator **emplace** (_Args &&... __args)
- template<typename... _Args>
  iterator **emplace_hint** (const_iterator __hint, _Args &&... __args)
- const_iterator **end** () const noexcept
- iterator **end** () noexcept
- local_iterator **end** (size_type __b)
- const_local_iterator **end** (size_type __b) const
- template<typename _Kt, typename = std::__has_is_transparent_t<_Hash, _Kt>, typename = std::__has_is_transparent_t<_Pred, _Kt>>
  std::pair< iterator, iterator > **equal_range** (const _Kt &__k)
- template<typename _Kt, typename = std::__has_is_transparent_t<_Hash, _Kt>, typename = std::__has_is_transparent_t<_Pred, _Kt>>
  std::pair< const_iterator, const_iterator > **equal_range** (const _Kt &__k) const
- std::pair< iterator, iterator > **equal_range** (const key_type &__key)
- std::pair< const_iterator, const_iterator > **equal_range** (const key_type &__key) const
- _Base_iterator **erase** (_Base_const_iterator __it)
- size_type **erase** (const key_type &__key)
- iterator **erase** (const_iterator __first, const_iterator __last)
- iterator **erase** (const_iterator __it)
- iterator **erase** (iterator __it)
- node_type **extract** (const key_type &__key)
- node_type **extract** (const_iterator __position)
- template<typename _Kt, typename = std::__has_is_transparent_t<_Hash, _Kt>, typename = std::__has_is_transparent_t<_Pred, _Kt>>
  iterator **find** (const _Kt &__k)
- template<typename _Kt, typename = std::__has_is_transparent_t<_Hash, _Kt>, typename = std::__has_is_transparent_t<_Pred, _Kt>>
  const_iterator **find** (const _Kt &__k) const

- iterator **find** (const key_type &__key)
- const_iterator **find** (const key_type &__key) const
- template<typename _InputIterator>
  void **insert** (_InputIterator __first, _InputIterator __last)
- iterator **insert** (const value_type &__obj)
- iterator **insert** (const_iterator __hint, const value_type &__obj)
- iterator **insert** (const_iterator __hint, node_type &&__nh)
- iterator **insert** (const_iterator __hint, value_type &&__obj)
- iterator **insert** (node_type &&__nh)
- void **insert** (std::initializer_list< value_type > __l)
- iterator **insert** (value_type &&__obj)
- float **max_load_factor** () const noexcept
- void **max_load_factor** (float __f)
- template<typename _H2, typename _P2>
  void **merge** (unordered_multiset< _Value, _H2, _P2, _Alloc > &&__source)
- template<typename _H2, typename _P2>
  void **merge** (unordered_multiset< _Value, _H2, _P2, _Alloc > &__source)
- template<typename _H2, typename _P2>
  void **merge** (unordered_set< _Value, _H2, _P2, _Alloc > &&__source)
- template<typename _H2, typename _P2>
  void **merge** (unordered_set< _Value, _H2, _P2, _Alloc > &__source)
- unordered_multiset & **operator=** (const unordered_multiset &)=default
- unordered_multiset & **operator=** (initializer_list< value_type > __l)
- unordered_multiset & **operator=** (unordered_multiset &&)=default
- void **swap** (unordered_multiset &__x) noexcept(noexcept(declval< _Base & >().swap(__x)))

Protected Member Functions

- constexpr void **_M_swap** (const _Safe_container &__x) const noexcept

Friends

- template<typename _ItT, typename _SeqT, typename _CatT>
  class **::__gnu_debug::_Safe_iterator**
- template<typename _ItT, typename _SeqT>
  class **::__gnu_debug::_Safe_local_iterator**

5.999.1 Detailed Description

template<typename _Value, typename _Hash = std::hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>>
class std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >

Class std::unordered_multiset with safety/checking/debug instrumentation.
The documentation for this class was generated from the following file:

- debug/unordered_set

5.1000 std::unordered_multiset< _Value, _Hash, _Pred, _Alloc > Class Template Reference

`#include <unordered_set>`

Public Types

- typedef _Hashtable::key_type key_type
- typedef _Hashtable::value_type value_type
- typedef _Hashtable::hasher hasher
- typedef _Hashtable::key_equal key_equal
- typedef _Hashtable::allocator_type allocator_type

- typedef _Hashtable::pointer pointer
- typedef _Hashtable::const_pointer const_pointer
- typedef _Hashtable::reference reference
- typedef _Hashtable::const_reference const_reference
- typedef _Hashtable::iterator iterator
- typedef _Hashtable::const_iterator const_iterator
- typedef _Hashtable::local_iterator local_iterator
- typedef _Hashtable::const_local_iterator const_local_iterator
- typedef _Hashtable::size_type size_type
- typedef _Hashtable::difference_type difference_type

Public Member Functions

- unordered_multiset ()=default
- template<typename _InputIterator>
  unordered_multiset (_InputIterator __first, _InputIterator __last, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_multiset (_InputIterator __first, _InputIterator __last, size_type __n, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_multiset (_InputIterator __first, _InputIterator __last, size_type __n, const hasher &__hf, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_multiset (_InputIterator __first, _InputIterator __last, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- unordered_multiset (const allocator_type &__a)
- unordered_multiset (const unordered_multiset &)=default
- **unordered_multiset** (const unordered_multiset &__umset, const allocator_type &__a)
- **unordered_multiset** (initializer_list< value_type > __l, const allocator_type &__a)
- **unordered_multiset** (initializer_list< value_type > __l, size_type __n, const allocator_type &__a)
- **unordered_multiset** (initializer_list< value_type > __l, size_type __n, const hasher &__hf, const allocator_type &__a)
- unordered_multiset (initializer_list< value_type > __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_multiset** (size_type __n, const allocator_type &__a)
- **unordered_multiset** (size_type __n, const hasher &__hf, const allocator_type &__a)
- unordered_multiset (size_type __n, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- unordered_multiset (unordered_multiset &&)=default
- **unordered_multiset** (unordered_multiset &&__umset, const allocator_type &__a) noexcept(noexcept(_↩ Hashtable(std::move(__umset._M_h), __a)))
- size_type **bucket** (const key_type &__key) const
- size_type bucket_count () const noexcept

- size_type **bucket_size** (size_type __n) const
- const_iterator cbegin () const noexcept
- const_iterator cend () const noexcept
- void clear () noexcept
- template<typename... _Args>
  iterator emplace (_Args &&... __args)
- template<typename... _Args>
  iterator emplace_hint (const_iterator __pos, _Args &&... __args)
- bool empty () const noexcept
- size_type erase (const key_type &__x)
- iterator erase (const_iterator __first, const_iterator __last)
- allocator_type get_allocator () const noexcept
- hasher hash_function () const
- template<typename _InputIterator>
  void insert (_InputIterator __first, _InputIterator __last)
- void insert (initializer_list< value_type > __l)
- key_equal key_eq () const
- float load_factor () const noexcept
- size_type max_bucket_count () const noexcept
- float max_load_factor () const noexcept
- void max_load_factor (float __z)
- size_type max_size () const noexcept
- unordered_multiset & operator= (const unordered_multiset &)=default
- unordered_multiset & operator= (initializer_list< value_type > __l)
- unordered_multiset & operator= (unordered_multiset &&)=default
- void rehash (size_type __n)
- void reserve (size_type __n)
- size_type size () const noexcept
- void swap (unordered_multiset &__x) noexcept(noexcept(_M_h.swap(__x._M_h)))

- iterator begin () noexcept
- const_iterator begin () const noexcept

- iterator end () noexcept
- const_iterator end () const noexcept

- iterator insert (const value_type &__x)
- iterator insert (value_type &&__x)

- iterator insert (const_iterator __hint, const value_type &__x)
- iterator insert (const_iterator __hint, value_type &&__x)

- iterator erase (const_iterator __position)
- iterator erase (iterator __position)

- iterator find (const key_type &__x)
- template<typename _Kt>
  auto find (const _Kt &__x) -> decltype(_M_h._M_find_tr(__x))
- const_iterator find (const key_type &__x) const
- template<typename _Kt>
  auto find (const _Kt &__x) const -> decltype(_M_h._M_find_tr(__x))

- size_type count (const key_type &__x) const
- template<typename _Kt>
  auto count (const _Kt &__x) const -> decltype(_M_h._M_count_tr(__x))

- bool contains (const key_type &__x) const
- template<typename _Kt>
  auto contains (const _Kt &__x) const -> decltype(_M_h._M_find_tr(__x), void(), true)

- std::pair< iterator, iterator > equal_range (const key_type &__x)
- template<typename _Kt>
  auto equal_range (const _Kt &__x) -> decltype(_M_h._M_equal_range_tr(__x))
- std::pair< const_iterator, const_iterator > equal_range (const key_type &__x) const
- template<typename _Kt>
  auto equal_range (const _Kt &__x) const -> decltype(_M_h._M_equal_range_tr(__x))

- local_iterator begin (size_type __n)
- const_local_iterator begin (size_type __n) const
- const_local_iterator cbegin (size_type __n) const

- local_iterator end (size_type __n)
- const_local_iterator end (size_type __n) const
- const_local_iterator cend (size_type __n) const

Friends

- template<typename _Value1, typename _Hash1, typename _Pred1, typename _Alloc1>
  bool **operator==** (const unordered_multiset< _Value1, _Hash1, _Pred1, _Alloc1 > &, const unordered_multiset< _Value1, _Hash1, _Pred1, _Alloc1 > &)

5.1000.1 Detailed Description

template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>, typename _Alloc = allocator<_Value>>
class std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >

A standard container composed of equivalent keys (possibly containing multiple of each key value) in which the elements' keys are the elements themselves.

Since

C++11

Template Parameters

_Value	Type of key objects.
_Hash	Hashing function object type, defaults to hash<_Value>.
_Pred	Predicate function object type, defaults to equal_to<_Value>.
_Alloc	Allocator type, defaults to allocator<_Key>.

Meets the requirements of a container, and unordered associative container
Base is _Hashtable, dispatched at compile time via template alias __umset_hashtable.

5.1000.2 Member Typedef Documentation

allocator_type

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::allocator_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >↩
::allocator_type
```

Public typedefs.

const_iterator

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >↩
::const_iterator
```

Iterator-related typedefs.

const_local_iterator

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc
>::const_local_iterator
```

Iterator-related typedefs.

const_pointer

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_pointer std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::const↩
_pointer
```

Iterator-related typedefs.

const_reference

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_reference std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >↩
::const_reference
```

Iterator-related typedefs.

difference_type

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::difference_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >↩
::difference_type
```

Iterator-related typedefs.

hasher

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::hasher std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::hasher
```

Public typedefs.

iterator

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::iterator
```

Iterator-related typedefs.

key_equal

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::key_equal std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::key_equal
```

Public typedefs.

key_type

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::key_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::key_type
```

Public typedefs.

local_iterator

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >↩
::local_iterator
```

Iterator-related typedefs.

pointer

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::pointer std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::pointer
```

Iterator-related typedefs.

reference

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::reference std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::reference
```

Iterator-related typedefs.

size_type

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::size_type
```

Iterator-related typedefs.

value_type

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::value_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::value_↩
type
```

Public typedefs.

5.1000.3 Constructor & Destructor Documentation

unordered_multiset() [1/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset ()  [default]
```

Default constructor.

unordered_multiset() [2/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
            size_type __n,
            const hasher & __hf = hasher(),
            const key_equal & __eql = key_equal(),
            const allocator_type & __a = allocator_type())  [inline], [explicit]
```

Default constructor creates no elements.

Parameters

__n	Minimal initial number of buckets.
__hf	A hash functor.
__eql	A key equality functor.
__a	An allocator object.

unordered_multiset() [3/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _InputIterator>
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
            _InputIterator __first,
            _InputIterator __last,
            size_type __n = 0,
            const hasher & __hf = hasher(),
```

```
            const key_equal & __eql = key_equal(),
            const allocator_type & __a = allocator_type())  [inline]
```

Builds an unordered_multiset from a range.

Parameters

__first	An input iterator.
__last	An input iterator.
__n	Minimal initial number of buckets.
__hf	A hash functor.
__eql	A key equality functor.
__a	An allocator object.

Create an unordered_multiset consisting of copies of the elements from [__first,__last). This is linear in N (where N is distance(__first,__last)).

unordered_multiset() [4/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
            const unordered_multiset< _Value, _Hash, _Pred, _Alloc > & )  [default]
```

Copy constructor.

unordered_multiset() [5/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
            unordered_multiset< _Value, _Hash, _Pred, _Alloc > && )  [default]
```

Move constructor.

unordered_multiset() [6/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
            initializer_list< value_type > __l,
            size_type __n = 0,
            const hasher & __hf = hasher(),
            const key_equal & __eql = key_equal(),
            const allocator_type & __a = allocator_type())  [inline]
```

Builds an unordered_multiset from an initializer_list.

Parameters

__l	An initializer_list.
__n	Minimal initial number of buckets.
__hf	A hash functor.
__eql	A key equality functor.
__a	An allocator object.

Create an unordered_multiset consisting of copies of the elements in the list. This is linear in N (where N is *__l.size()*).

unordered_multiset() [7/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
            const allocator_type & __a)  [inline], [explicit]
```

Creates an unordered_multiset with no elements.

Parameters

_↩ _a	An allocator object.

5.1000.4 Member Function Documentation

begin() [1/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::begin () const  [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered_multiset.

begin() [2/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::begin ()  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered_multiset.

begin() [3/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::begin (
            size_type __n)  [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

_↩ _n	The bucket index.

Returns

A read-only local iterator.

begin() [4/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::begin (
            size_type __n) const  [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

_↩ _n	The bucket index.

Returns

A read-only local iterator.

bucket_count()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::bucket_count () const  [inline],
[noexcept]
```

Returns the number of buckets of the unordered_multiset.

cbegin() [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::cbegin () const  [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered_multiset.

cbegin() [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::cbegin (
            size_type __n) const  [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

_↩ _n	The bucket index.

Returns

A read-only local iterator.

cend() [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::cend () const  [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered_multiset.

cend() [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::cend (
            size_type __n) const  [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

_↩ _n	The bucket index.

Returns

A read-only local iterator.

clear()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::clear ()  [inline], [noexcept]
```

Erases all elements in an unordered_multiset.
Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

contains() [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _Kt>
auto std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::contains (
            const _Kt & __x) const -> decltype(_M_h._M_find_tr(__x), void(), true)   [inline]
```

Finds whether an element with the given key exists.

Parameters

_↩ _x	Key of elements to be located.

Returns

True if there is any element with the specified key.

contains() [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
bool std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::contains (
            const key_type & __x) const  [inline]
```

Finds whether an element with the given key exists.

Parameters

_↩ _x	Key of elements to be located.

Returns

True if there is any element with the specified key.

count() [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _Kt>
auto std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::count (
            const _Kt & __x) const -> decltype(_M_h._M_count_tr(__x))   [inline]
```

Finds the number of elements.

Parameters

_↩ _x	Element to located.

Returns

Number of elements with specified key.

count() [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::count (
            const key_type & __x) const  [inline]
```

Finds the number of elements.

Parameters

_↩ _x	Element to located.

Returns

Number of elements with specified key.

emplace()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename... _Args>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::emplace (
            _Args &&... __args)  [inline]
```

Builds and insert an element into the unordered_multiset.

Parameters

__args	Arguments used to generate an element.

Returns

An iterator that points to the inserted element.

Insertion requires amortized constant time.

emplace_hint()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename... _Args>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::emplace_hint (
            const_iterator __pos,
            _Args &&... __args)  [inline]
```

Inserts an element into the unordered_multiset.

Parameters

__pos	An iterator that serves as a hint as to where the element should be inserted.
__args	Arguments used to generate the element to be inserted.

Returns

An iterator that points to the inserted element.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
For more on *hinting*, see: https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints
Insertion requires amortized constant time.

empty()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
bool std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::empty () const  [inline], [nodiscard],
[noexcept]
```

Returns true if the unordered_multiset is empty.

end() [1/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::end () const  [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered_multiset.

end() [2/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::end ()  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered_multiset.

end() [3/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::end (
            size_type __n)  [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

_↩ _n	The bucket index.

Returns

A read-only local iterator.

end() [4/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::end (
            size_type __n) const  [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

_↩ _n	The bucket index.

Returns

A read-only local iterator.

equal_range() [1/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _Kt>
auto std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::equal_range (
            const _Kt & __x) -> decltype(_M_h._M_equal_range_tr(__x))   [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

equal_range() [2/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _Kt>
auto std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::equal_range (
            const _Kt & __x) const -> decltype(_M_h._M_equal_range_tr(__x))   [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

equal_range() [3/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::pair< iterator, iterator > std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::equal↩
_range (
            const key_type & __x)  [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

equal_range() [4/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::pair< const_iterator, const_iterator > std::unordered_multiset< _Value, _Hash, _Pred, _Alloc
>::equal_range (
            const key_type & __x) const  [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

erase() [1/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::erase (
            const key_type & __x)  [inline]
```

Erases elements according to the provided key.

Parameters

_↩ _x	Key of element to be erased.

Returns

The number of elements erased.

This function erases all the elements located by the given key from an unordered_multiset.
Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [2/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::erase (
            const_iterator __first,
            const_iterator __last)  [inline]
```

Erases a [__first,__last) range of elements from an unordered_multiset.

Parameters

__first	Iterator pointing to the start of the range to be erased.
__last	Iterator pointing to the end of the range to be erased.

Returns

The iterator *__last*.

This function erases a sequence of elements from an unordered_multiset.
Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [3/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::erase (
            const_iterator __position)  [inline]
```

Erases an element from an unordered_multiset.

Parameters

__position	An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *__position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from an unordered_multiset.
Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [4/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::erase (
            iterator __position)  [inline]
```

Erases an element from an unordered_multiset.

Parameters

__position	An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *__position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from an unordered_multiset.
Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

find() [1/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _Kt>
auto std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::find (
            const _Kt & __x) -> decltype(_M_h._M_find_tr(__x))   [inline]
```

Tries to locate an element in an unordered_multiset.

Parameters

_↩ _x	Element to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

find() [2/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _Kt>
```

```
auto std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::find (
            const _Kt & __x) const -> decltype(_M_h._M_find_tr(__x))   [inline]
```

Tries to locate an element in an unordered_multiset.

Parameters

_↩ _x	Element to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

find() [3/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::find (
            const key_type & __x)  [inline]
```

Tries to locate an element in an unordered_multiset.

Parameters

_↩ _x	Element to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

find() [4/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::find (
            const key_type & __x) const  [inline]
```

Tries to locate an element in an unordered_multiset.

Parameters

_↩ _x	Element to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

get_allocator()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
allocator_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::get_allocator () const
[inline], [noexcept]
```

Returns the allocator object used by the unordered_multiset.

hash_function()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
hasher std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::hash_function () const  [inline]
```

Returns the hash functor object with which the unordered_multiset was constructed.

insert() [1/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _InputIterator>
void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (
            _InputIterator __first,
            _InputIterator __last)  [inline]
```

A template function that inserts a range of elements.

Parameters

__first	Iterator pointing to the start of the range to be inserted.
__last	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

insert() [2/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (
            const value_type & __x)  [inline]
```

Inserts an element into the unordered_multiset.

Parameters

_↩ *_x*	Element to be inserted.

Returns

An iterator that points to the inserted element.

Insertion requires amortized constant time.

insert() [3/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (
            const_iterator __hint,
            const value_type & __x)  [inline]
```

Inserts an element into the unordered_multiset.

Parameters

__hint	An iterator that serves as a hint as to where the element should be inserted.
__x	Element to be inserted.

Returns

An iterator that points to the inserted element.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
For more on *hinting*, see: https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints
Insertion requires amortized constant.

insert() [4/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (
            const_iterator __hint,
            value_type && __x)  [inline]
```

Inserts an element into the unordered_multiset.

Parameters

__hint	An iterator that serves as a hint as to where the element should be inserted.
__x	Element to be inserted.

Returns

An iterator that points to the inserted element.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
For more on *hinting*, see: https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints
Insertion requires amortized constant.

insert() [5/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (
            initializer_list< value_type > __l)  [inline]
```

Inserts a list of elements into the unordered_multiset.

Parameters

↩ _↩ ↩ _↩ *l*	A std::initializer_list<value_type> of elements to be inserted.

Complexity similar to that of the range constructor.

insert() [6/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (
            value_type && __x)  [inline]
```

Inserts an element into the unordered_multiset.

Parameters

_↩ *_x*	Element to be inserted.

Returns

An iterator that points to the inserted element.

Insertion requires amortized constant time.

key_eq()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
key_equal std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::key_eq () const  [inline]
```

Returns the key comparison object with which the unordered_multiset was constructed.

load_factor()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
float std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::load_factor () const  [inline],
[noexcept]
```

Returns the average number of elements per bucket.

max_bucket_count()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::max_bucket_count () const
[inline], [noexcept]
```

Returns the maximum number of buckets of the unordered_multiset.

max_load_factor() [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
float std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::max_load_factor () const  [inline],
[noexcept]
```

Returns a positive number that the unordered_multiset tries to keep the load factor less than or equal to.

max_load_factor() [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::max_load_factor (
            float __z)  [inline]
```

Change the unordered_multiset maximum load factor.

Parameters

_↩ _z	The new maximum load factor.

max_size()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::max_size () const  [inline],
[noexcept]
```

Returns the maximum size of the unordered_multiset.

operator=() [1/3]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
unordered_multiset & std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::operator= (
            const unordered_multiset< _Value, _Hash, _Pred, _Alloc > & )  [default]
```

Copy assignment operator.

operator=() [2/3]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
unordered_multiset & std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::operator= (
            initializer_list< value_type > __l)  [inline]
```

Unordered_multiset list assignment operator.

Parameters

↩ _↩ ↩ _↩ l	An initializer_list.

This function fills an unordered_multiset with copies of the elements in the initializer list __l.

Note that the assignment completely changes the unordered_multiset and that the resulting unordered_multiset's size is the same as the number of elements assigned.

operator=() [3/3]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
unordered_multiset & std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::operator= (
            unordered_multiset< _Value, _Hash, _Pred, _Alloc > && )  [default]
```

Move assignment operator.

rehash()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::rehash (
            size_type __n)  [inline]
```

May rehash the unordered_multiset.

Parameters

_↩ _n	The new number of buckets.

Rehash will occur only if the new number of buckets respect the unordered_multiset maximum load factor.

reserve()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::reserve (
            size_type __n)  [inline]
```

Prepare the unordered_multiset for a specified number of elements.

Parameters

_↩ _n	Number of elements required.

Same as rehash(ceil(n / max_load_factor())).

size()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::size () const  [inline], [noexcept]
```

Returns the size of the unordered_multiset.

swap()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::swap (
            unordered_multiset< _Value, _Hash, _Pred, _Alloc > & __x)  [inline], [noexcept]
```

Swaps data with another unordered_multiset.

Parameters

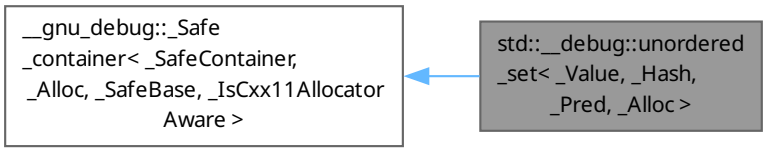
_↩ _x	An unordered_multiset of the same element and allocator types.

This exchanges the elements between two sets in constant time. Note that the global std::swap() function is specialized such that std::swap(s1,s2) will feed to this function.
The documentation for this class was generated from the following file:

- unordered_set.h

5.1001 std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc > Class Template Reference

`#include <unordered_set>`
Inheritance diagram for std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc >:



Public Types

- typedef _Base::allocator_type **allocator_type**
- typedef __gnu_debug::_Safe_iterator< _Base_const_iterator, unordered_set > **const_iterator**
- typedef __gnu_debug::_Safe_local_iterator< _Base_const_local_iterator, unordered_set > **const_local_↩ iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**
- typedef _Base::difference_type **difference_type**
- typedef _Base::hasher **hasher**
- using **insert_return_type**
- typedef __gnu_debug::_Safe_iterator< _Base_iterator, unordered_set > **iterator**
- typedef _Base::key_equal **key_equal**
- typedef _Base::key_type **key_type**
- typedef __gnu_debug::_Safe_local_iterator< _Base_local_iterator, unordered_set > **local_iterator**
- using **node_type**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef _Base::size_type **size_type**
- typedef _Base::value_type **value_type**

Public Member Functions

- **unordered_set** (_Base_ref __x)
- template<typename _InputIterator>
  unordered_set (_InputIterator __first, _InputIterator __last, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_set (_InputIterator __first, _InputIterator __last, size_type __n, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_set (_InputIterator __first, _InputIterator __last, size_type __n, const hasher &__hf, const allocator↩ _type &__a)
- template<typename _InputIterator>
  unordered_set (_InputIterator __first, _InputIterator __last, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_set** (const allocator_type &__a)
- **unordered_set** (const unordered_set &)=default
- **unordered_set** (const unordered_set &__uset, const allocator_type &__a)
- **unordered_set** (initializer_list< value_type > __l, const allocator_type &__a)
- **unordered_set** (initializer_list< value_type > __l, size_type __n, const allocator_type &__a)
- **unordered_set** (initializer_list< value_type > __l, size_type __n, const hasher &__hf, const allocator_type &__a)
- **unordered_set** (initializer_list< value_type > __l, size_type __n=0, const hasher &__hf=hasher(), const key_↩ equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_set** (size_type __n, const allocator_type &__a)
- **unordered_set** (size_type __n, const hasher &__hf, const allocator_type &__a)
- **unordered_set** (size_type __n, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_set** (unordered_set &&)=default
- **unordered_set** (unordered_set &&__uset, const allocator_type &__a) noexcept(noexcept(_Base(std::move(_↩ _uset), __a)))
- const _Base & **_M_base** () const noexcept
- _Base & **_M_base** () noexcept
- const_iterator **begin** () const noexcept
- iterator **begin** () noexcept
- local_iterator **begin** (size_type __b)
- const_local_iterator **begin** (size_type __b) const
- size_type **bucket_size** (size_type __b) const
- const_iterator **cbegin** () const noexcept
- const_local_iterator **cbegin** (size_type __b) const
- const_iterator **cend** () const noexcept
- const_local_iterator **cend** (size_type __b) const
- void **clear** () noexcept
- template<typename... _Args>
  std::pair< iterator, bool > **emplace** (_Args &&... __args)
- template<typename... _Args>
  iterator **emplace_hint** (const_iterator __hint, _Args &&... __args)
- const_iterator **end** () const noexcept
- iterator **end** () noexcept
- local_iterator **end** (size_type __b)
- const_local_iterator **end** (size_type __b) const
- template<typename _Kt, typename = std::__has_is_transparent_t<_Hash, _Kt>, typename = std::__has_is_transparent_t<_Pred, _Kt>>
  std::pair< iterator, iterator > **equal_range** (const _Kt &__k)
- template<typename _Kt, typename = std::__has_is_transparent_t<_Hash, _Kt>, typename = std::__has_is_transparent_t<_Pred, _Kt>>
  std::pair< const_iterator, const_iterator > **equal_range** (const _Kt &__k) const

- std::pair< iterator, iterator > **equal_range** (const key_type &__key)
- std::pair< const_iterator, const_iterator > **equal_range** (const key_type &__key) const
- _Base_iterator **erase** (_Base_const_iterator __it)
- size_type **erase** (const key_type &__key)
- iterator **erase** (const_iterator __first, const_iterator __last)
- iterator **erase** (const_iterator __it)
- iterator **erase** (iterator __it)
- node_type **extract** (const key_type &__key)
- node_type **extract** (const_iterator __position)
- template<typename _Kt, typename = std::__has_is_transparent_t<_Hash, _Kt>, typename = std::__has_is_transparent_t<_Pred, _Kt>>
  iterator **find** (const _Kt &__k)
- template<typename _Kt, typename = std::__has_is_transparent_t<_Hash, _Kt>, typename = std::__has_is_transparent_t<_Pred, _Kt>>
  const_iterator **find** (const _Kt &__k) const
- iterator **find** (const key_type &__key)
- const_iterator **find** (const key_type &__key) const
- template<typename _InputIterator>
  void **insert** (_InputIterator __first, _InputIterator __last)
- std::pair< iterator, bool > **insert** (const value_type &__obj)
- iterator **insert** (const_iterator __hint, const value_type &__obj)
- iterator **insert** (const_iterator __hint, node_type &&__nh)
- iterator **insert** (const_iterator __hint, value_type &&__obj)
- insert_return_type **insert** (node_type &&__nh)
- void **insert** (std::initializer_list< value_type > __l)
- std::pair< iterator, bool > **insert** (value_type &&__obj)
- float **max_load_factor** () const noexcept
- void **max_load_factor** (float __f)
- template<typename _H2, typename _P2>
  void **merge** (unordered_multiset< _Value, _H2, _P2, _Alloc > &&__source)
- template<typename _H2, typename _P2>
  void **merge** (unordered_multiset< _Value, _H2, _P2, _Alloc > &__source)
- template<typename _H2, typename _P2>
  void **merge** (unordered_set< _Value, _H2, _P2, _Alloc > &&__source)
- template<typename _H2, typename _P2>
  void **merge** (unordered_set< _Value, _H2, _P2, _Alloc > &__source)
- unordered_set & **operator=** (const unordered_set &)=default
- unordered_set & **operator=** (initializer_list< value_type > __l)
- unordered_set & **operator=** (unordered_set &&)=default
- void **swap** (unordered_set &__x) noexcept(noexcept(declval< _Base & >().swap(__x)))

Protected Member Functions

- constexpr void **_M_swap** (const _Safe_container &__x) const noexcept

Friends

- template<typename _ItT, typename _SeqT, typename _CatT>
  class **::__gnu_debug::_Safe_iterator**
- template<typename _ItT, typename _SeqT>
  class **::__gnu_debug::_Safe_local_iterator**

5.1001.1 Detailed Description

template<typename _Value, typename _Hash = std::hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>>
class std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc >

Class std::unordered_set with safety/checking/debug instrumentation.
The documentation for this class was generated from the following file:

- debug/unordered_set

5.1002 std::unordered_set< _Value, _Hash, _Pred, _Alloc > Class Template Reference

`#include <unordered_set>`

Public Types

- typedef _Hashtable::key_type key_type
- typedef _Hashtable::value_type value_type
- typedef _Hashtable::hasher hasher
- typedef _Hashtable::key_equal key_equal
- typedef _Hashtable::allocator_type allocator_type

- typedef _Hashtable::pointer pointer
- typedef _Hashtable::const_pointer const_pointer
- typedef _Hashtable::reference reference
- typedef _Hashtable::const_reference const_reference
- typedef _Hashtable::iterator iterator
- typedef _Hashtable::const_iterator const_iterator
- typedef _Hashtable::local_iterator local_iterator
- typedef _Hashtable::const_local_iterator const_local_iterator
- typedef _Hashtable::size_type size_type
- typedef _Hashtable::difference_type difference_type

Public Member Functions

- unordered_set ()=default
- template<typename _InputIterator>
  unordered_set (_InputIterator __first, _InputIterator __last, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_set (_InputIterator __first, _InputIterator __last, size_type __n, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_set (_InputIterator __first, _InputIterator __last, size_type __n, const hasher &__hf, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_set (_InputIterator __first, _InputIterator __last, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- unordered_set (const allocator_type &__a)
- unordered_set (const unordered_set &)=default
- **unordered_set** (const unordered_set &__uset, const allocator_type &__a)
- **unordered_set** (initializer_list< value_type > __l, const allocator_type &__a)
- **unordered_set** (initializer_list< value_type > __l, size_type __n, const allocator_type &__a)

- **unordered_set** (initializer_list< value_type > __l, size_type __n, const hasher &__hf, const allocator_type &__a)
- unordered_set (initializer_list< value_type > __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_set** (size_type __n, const allocator_type &__a)
- **unordered_set** (size_type __n, const hasher &__hf, const allocator_type &__a)
- unordered_set (size_type __n, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- unordered_set (unordered_set &&)=default
- **unordered_set** (unordered_set &&__uset, const allocator_type &__a) noexcept(noexcept(_Hashtable(std::move(↩ __uset._M_h), __a)))
- size_type **bucket** (const key_type &__key) const
- size_type bucket_count () const noexcept
- size_type **bucket_size** (size_type __n) const
- const_iterator cbegin () const noexcept
- const_iterator cend () const noexcept
- void clear () noexcept
- template<typename... _Args>
  std::pair< iterator, bool > emplace (_Args &&... __args)
- template<typename... _Args>
  iterator emplace_hint (const_iterator __pos, _Args &&... __args)
- bool empty () const noexcept
- size_type erase (const key_type &__x)
- iterator erase (const_iterator __first, const_iterator __last)
- allocator_type get_allocator () const noexcept
- hasher hash_function () const
- template<typename _InputIterator>
  void insert (_InputIterator __first, _InputIterator __last)
- void insert (initializer_list< value_type > __l)
- key_equal key_eq () const
- float load_factor () const noexcept
- size_type max_bucket_count () const noexcept
- float max_load_factor () const noexcept
- void max_load_factor (float __z)
- size_type max_size () const noexcept
- unordered_set & operator= (const unordered_set &)=default
- unordered_set & operator= (initializer_list< value_type > __l)
- unordered_set & operator= (unordered_set &&)=default
- void rehash (size_type __n)
- void reserve (size_type __n)
- size_type size () const noexcept
- void swap (unordered_set &__x) noexcept(noexcept(_M_h.swap(__x._M_h)))

- iterator begin () noexcept
- const_iterator begin () const noexcept

- iterator end () noexcept
- const_iterator end () const noexcept

- std::pair< iterator, bool > insert (const value_type &__x)
- std::pair< iterator, bool > insert (value_type &&__x)

- iterator insert (const_iterator __hint, const value_type &__x)
- iterator insert (const_iterator __hint, value_type &&__x)

- iterator erase (const_iterator __position)
- iterator erase (iterator __position)

- iterator find (const key_type &__x)
- template<typename _Kt>
  auto find (const _Kt &__k) -> decltype(_M_h._M_find_tr(__k))
- const_iterator find (const key_type &__x) const
- template<typename _Kt>
  auto find (const _Kt &__k) const -> decltype(_M_h._M_find_tr(__k))

- size_type count (const key_type &__x) const
- template<typename _Kt>
  auto count (const _Kt &__k) const -> decltype(_M_h._M_count_tr(__k))

- bool contains (const key_type &__x) const
- template<typename _Kt>
  auto contains (const _Kt &__k) const -> decltype(_M_h._M_find_tr(__k), void(), true)

- std::pair< iterator, iterator > equal_range (const key_type &__x)
- template<typename _Kt>
  auto equal_range (const _Kt &__k) -> decltype(_M_h._M_equal_range_tr(__k))
- std::pair< const_iterator, const_iterator > equal_range (const key_type &__x) const
- template<typename _Kt>
  auto equal_range (const _Kt &__k) const -> decltype(_M_h._M_equal_range_tr(__k))

- local_iterator begin (size_type __n)
- const_local_iterator begin (size_type __n) const
- const_local_iterator cbegin (size_type __n) const

- local_iterator end (size_type __n)
- const_local_iterator end (size_type __n) const
- const_local_iterator cend (size_type __n) const

Friends

- template<typename _Value1, typename _Hash1, typename _Pred1, typename _Alloc1>
  bool **operator==** (const unordered_set< _Value1, _Hash1, _Pred1, _Alloc1 > &, const unordered_set< _Value1, _Hash1, _Pred1, _Alloc1 > &)

5.1002.1 Detailed Description

template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>, typename _Alloc = allocator<_Value>>
class std::unordered_set< _Value, _Hash, _Pred, _Alloc >

A standard container composed of unique keys (containing at most one of each key value) in which the elements' keys are the elements themselves.

Since

C++11

Template Parameters

_Value	Type of key objects.
_Hash	Hashing function object type, defaults to hash<_Value>.
_Pred	Predicate function object type, defaults to equal_to<_Value>.
_Alloc	Allocator type, defaults to allocator<_Key>.

Meets the requirements of a container, and unordered associative container
Base is _Hashtable, dispatched at compile time via template alias __uset_hashtable.

5.1002.2 Member Typedef Documentation

allocator_type

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::allocator_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::allocator↩
_type
```

Public typedefs.

const_iterator

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::const_↩
iterator
```

Iterator-related typedefs.

const_local_iterator

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >↩
::const_local_iterator
```

Iterator-related typedefs.

const_pointer

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_pointer std::unordered_set< _Value, _Hash, _Pred, _Alloc >::const_↩
pointer
```

Iterator-related typedefs.

const_reference

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_reference std::unordered_set< _Value, _Hash, _Pred, _Alloc >::const_↩
reference
```

Iterator-related typedefs.

difference_type

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::difference_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::difference↩
_type
```

Iterator-related typedefs.

hasher

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::hasher std::unordered_set< _Value, _Hash, _Pred, _Alloc >::hasher
```

Public typedefs.

iterator

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::iterator
```

Iterator-related typedefs.

key_equal

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::key_equal std::unordered_set< _Value, _Hash, _Pred, _Alloc >::key_equal
```

Public typedefs.

key_type

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::key_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::key_type
```

Public typedefs.

local_iterator

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
```

```
typedef _Hashtable::local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::local_↩
iterator
```
Iterator-related typedefs.

pointer

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::pointer std::unordered_set< _Value, _Hash, _Pred, _Alloc >::pointer
```
Iterator-related typedefs.

reference

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::reference std::unordered_set< _Value, _Hash, _Pred, _Alloc >::reference
```
Iterator-related typedefs.

size_type

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::size_type
```
Iterator-related typedefs.

value_type

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::value_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::value_type
```
Public typedefs.

5.1002.3 Constructor & Destructor Documentation

unordered_set() [1/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set ()  [default]
```
Default constructor.

unordered_set() [2/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (
            size_type __n,
            const hasher & __hf = hasher(),
            const key_equal & __eql = key_equal(),
            const allocator_type & __a = allocator_type())  [inline], [explicit]
```
Default constructor creates no elements.

Parameters

__n	Minimal initial number of buckets.
__hf	A hash functor.

Parameters

__eql	A key equality functor.
__a	An allocator object.

unordered_set() [3/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _InputIterator>
std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (
            _InputIterator __first,
            _InputIterator __last,
            size_type __n = 0,
            const hasher & __hf = hasher(),
            const key_equal & __eql = key_equal(),
            const allocator_type & __a = allocator_type())  [inline]
```

Builds an unordered_set from a range.

Parameters

__first	An input iterator.
__last	An input iterator.
__n	Minimal initial number of buckets.
__hf	A hash functor.
__eql	A key equality functor.
__a	An allocator object.

Create an unordered_set consisting of copies of the elements from [__first,__last). This is linear in N (where N is distance(__first,__last)).

unordered_set() [4/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (
            const unordered_set< _Value, _Hash, _Pred, _Alloc > & )  [default]
```

Copy constructor.

unordered_set() [5/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (
            unordered_set< _Value, _Hash, _Pred, _Alloc > && )  [default]
```

Move constructor.

unordered_set() [6/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
```

```
std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (
            const allocator_type & __a)  [inline], [explicit]
```

Creates an unordered_set with no elements.

Parameters

_↩ _a	An allocator object.

unordered_set() [7/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (
            initializer_list< value_type > __l,
            size_type __n = 0,
            const hasher & __hf = hasher(),
            const key_equal & __eql = key_equal(),
            const allocator_type & __a = allocator_type())  [inline]
```

Builds an unordered_set from an initializer_list.

Parameters

__l	An initializer_list.
__n	Minimal initial number of buckets.
__hf	A hash functor.
__eql	A key equality functor.
__a	An allocator object.

Create an unordered_set consisting of copies of the elements in the list. This is linear in N (where N is *__l.size()*).

5.1002.4 Member Function Documentation

begin() [1/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::begin () const  [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered_set.

begin() [2/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::begin ()  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered_set.

begin() [3/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::begin (
            size_type __n)  [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

_↩ _n	The bucket index.

Returns

A read-only local iterator.

begin() [4/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::begin (
            size_type __n) const  [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

_↩ _n	The bucket index.

Returns

A read-only local iterator.

bucket_count()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::bucket_count () const  [inline],
[noexcept]
```

Returns the number of buckets of the unordered_set.

cbegin() [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::cbegin () const  [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered_set.

cbegin() [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::cbegin (
            size_type __n) const  [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

_↩ _n	The bucket index.

Returns

A read-only local iterator.

cend() [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::cend () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered_set.

cend() [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::cend (
            size_type __n) const  [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

_↩ _n	The bucket index.

Returns

A read-only local iterator.

clear()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::clear ()  [inline], [noexcept]
```

Erases all elements in an unordered_set. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

contains() [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _Kt>
auto std::unordered_set< _Value, _Hash, _Pred, _Alloc >::contains (
            const _Kt & __k) const -> decltype(_M_h._M_find_tr(__k), void(), true)   [inline]
```

Finds whether an element with the given key exists.

Parameters

_↩ _x	Key of elements to be located.

Returns

True if there is any element with the specified key.

contains() [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
bool std::unordered_set< _Value, _Hash, _Pred, _Alloc >::contains (
            const key_type & __x) const  [inline]
```

Finds whether an element with the given key exists.

Parameters

_↩ _x	Key of elements to be located.

Returns

True if there is any element with the specified key.

count() [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _Kt>
auto std::unordered_set< _Value, _Hash, _Pred, _Alloc >::count (
            const _Kt & __k) const -> decltype(_M_h._M_count_tr(__k))   [inline]
```

Finds the number of elements.

Parameters

_↩ _x	Element to located.

Returns

Number of elements with specified key.

This function only makes sense for unordered_multisets; for unordered_set the result will either be 0 (not present) or 1 (present).

count() [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::count (
            const key_type & __x) const  [inline]
```

Finds the number of elements.

Parameters

_↩ _x	Element to located.

Returns

Number of elements with specified key.

This function only makes sense for unordered_multisets; for unordered_set the result will either be 0 (not present) or 1 (present).

emplace()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename... _Args>
std::pair< iterator, bool > std::unordered_set< _Value, _Hash, _Pred, _Alloc >::emplace (
            _Args &&... __args) [inline]
```

Attempts to build and insert an element into the unordered_set.

Parameters

__args	Arguments used to generate an element.

Returns

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to build and insert an element into the unordered_set. An unordered_set relies on unique keys and thus an element is only inserted if it is not already present in the unordered_set.
Insertion requires amortized constant time.

emplace_hint()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename... _Args>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::emplace_hint (
            const_iterator __pos,
            _Args &&... __args) [inline]
```

Attempts to insert an element into the unordered_set.

Parameters

__pos	An iterator that serves as a hint as to where the element should be inserted.
__args	Arguments used to generate the element to be inserted.

Returns

An iterator that points to the element with key equivalent to the one generated from *__args* (may or may not be the element itself).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument emplace() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
For more on *hinting*, see: https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints
Insertion requires amortized constant time.

empty()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
bool std::unordered_set< _Value, _Hash, _Pred, _Alloc >::empty () const [inline], [nodiscard],
[noexcept]
```

Returns true if the unordered_set is empty.

end() [1/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::end () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered_set.

end() [2/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::end ()  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered_set.

end() [3/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::end (
            size_type __n)  [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

_↩ _n	The bucket index.

Returns

A read-only local iterator.

end() [4/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::end (
            size_type __n) const  [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

_↩ _n	The bucket index.

Returns

A read-only local iterator.

equal_range() [1/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _Kt>
```

```
auto std::unordered_set< _Value, _Hash, _Pred, _Alloc >::equal_range (
            const _Kt & __k) -> decltype(_M_h._M_equal_range_tr(__k))   [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for multisets.

equal_range() [2/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _Kt>
auto std::unordered_set< _Value, _Hash, _Pred, _Alloc >::equal_range (
            const _Kt & __k) const -> decltype(_M_h._M_equal_range_tr(__k))   [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for multisets.

equal_range() [3/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::pair< iterator, iterator > std::unordered_set< _Value, _Hash, _Pred, _Alloc >::equal_range
(
            const key_type & __x)  [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for multisets.

equal_range() [4/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::pair< const_iterator, const_iterator > std::unordered_set< _Value, _Hash, _Pred, _Alloc >↩
::equal_range (
            const key_type & __x) const  [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _*x*	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for multisets.

erase() [1/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::erase (
            const key_type & __x)  [inline]
```

Erases elements according to the provided key.

Parameters

_↩ _*x*	Key of element to be erased.

Returns

The number of elements erased.

This function erases all the elements located by the given key from an unordered_set. For an unordered_set the result of this function can only be 0 (not present) or 1 (present). Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [2/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::erase (
            const_iterator __first,
            const_iterator __last)  [inline]
```

Erases a [__first,__last) range of elements from an unordered_set.

Parameters

__first	Iterator pointing to the start of the range to be erased.
__last	Iterator pointing to the end of the range to be erased.

Returns

The iterator *__last*.

This function erases a sequence of elements from an unordered_set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [3/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::erase (
            const_iterator __position)  [inline]
```

Erases an element from an unordered_set.

Parameters

__position	An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *__position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from an unordered_set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [4/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::erase (
            iterator __position)  [inline]
```

Erases an element from an unordered_set.

Parameters

__position	An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *__position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from an unordered_set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

find() [1/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _Kt>
auto std::unordered_set< _Value, _Hash, _Pred, _Alloc >::find (
            const _Kt & __k) -> decltype(_M_h._M_find_tr(__k))   [inline]
```

Tries to locate an element in an unordered_set.

Parameters

_x	Element to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

find() [2/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _Kt>
auto std::unordered_set< _Value, _Hash, _Pred, _Alloc >::find (
            const _Kt & __k) const -> decltype(_M_h._M_find_tr(__k))   [inline]
```

Tries to locate an element in an unordered_set.

Parameters

_↩ _x	Element to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

find() [3/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::find (
            const key_type & __x)  [inline]
```

Tries to locate an element in an unordered_set.

Parameters

_↩ _x	Element to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

find() [4/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::find (
            const key_type & __x) const  [inline]
```

Tries to locate an element in an unordered_set.

Parameters

_↩ _x	Element to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

get_allocator()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
allocator_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::get_allocator () const  [inline],
[noexcept]
```

Returns the allocator object used by the unordered_set.

hash_function()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
hasher std::unordered_set< _Value, _Hash, _Pred, _Alloc >::hash_function () const  [inline]
```

Returns the hash functor object with which the unordered_set was constructed.

insert() [1/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _InputIterator>
void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (
            _InputIterator __first,
            _InputIterator __last)  [inline]
```

A template function that attempts to insert a range of elements.

Parameters

__first	Iterator pointing to the start of the range to be inserted.
__last	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

insert() [2/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::pair< iterator, bool > std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (
            const value_type & __x)  [inline]
```

Attempts to insert an element into the unordered_set.

Parameters

_↩ _x	Element to be inserted.

Returns

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to insert an element into the unordered_set. An unordered_set relies on unique keys and thus an element is only inserted if it is not already present in the unordered_set.
Insertion requires amortized constant time.

insert() [3/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (
            const_iterator __hint,
            const value_type & __x)  [inline]
```

Attempts to insert an element into the unordered_set.

Parameters

__hint	An iterator that serves as a hint as to where the element should be inserted.
__x	Element to be inserted.

Returns

An iterator that points to the element with key of *__x* (may or may not be the element passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument insert() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
For more on *hinting*, see: https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.↩ html#containers.associative.insert_hints
Insertion requires amortized constant.

insert() [4/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (
            const_iterator __hint,
            value_type && __x)  [inline]
```

Attempts to insert an element into the unordered_set.

Parameters

__hint	An iterator that serves as a hint as to where the element should be inserted.
__x	Element to be inserted.

Returns

An iterator that points to the element with key of *__x* (may or may not be the element passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument insert() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
For more on *hinting*, see: https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints
Insertion requires amortized constant.

insert() [5/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (
            initializer_list< value_type > __l)  [inline]
```

Attempts to insert a list of elements into the unordered_set.

Parameters

__l	A std::initializer_list<value_type> of elements to be inserted.

Complexity similar to that of the range constructor.

insert() [6/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::pair< iterator, bool > std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (
            value_type && __x)  [inline]
```

Attempts to insert an element into the unordered_set.

Parameters

__x	Element to be inserted.

Returns

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to insert an element into the unordered_set. An unordered_set relies on unique keys and thus an element is only inserted if it is not already present in the unordered_set.
Insertion requires amortized constant time.

key_eq()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
key_equal std::unordered_set< _Value, _Hash, _Pred, _Alloc >::key_eq () const  [inline]
```

Returns the key comparison object with which the unordered_set was constructed.

load_factor()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
float std::unordered_set< _Value, _Hash, _Pred, _Alloc >::load_factor () const  [inline], [noexcept]
```

Returns the average number of elements per bucket.

max_bucket_count()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::max_bucket_count () const  [inline],
[noexcept]
```

Returns the maximum number of buckets of the unordered_set.

max_load_factor() [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
float std::unordered_set< _Value, _Hash, _Pred, _Alloc >::max_load_factor () const  [inline],
[noexcept]
```

Returns a positive number that the unordered_set tries to keep the load factor less than or equal to.

max_load_factor() [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::max_load_factor (
            float __z)  [inline]
```

Change the unordered_set maximum load factor.

Parameters

_↩ _z	The new maximum load factor.

max_size()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::max_size () const  [inline], [noexcept]
```

Returns the maximum size of the unordered_set.

operator=() [1/3]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
unordered_set & std::unordered_set< _Value, _Hash, _Pred, _Alloc >::operator= (
            const unordered_set< _Value, _Hash, _Pred, _Alloc > & )  [default]
```

Copy assignment operator.

operator=() [2/3]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
unordered_set & std::unordered_set< _Value, _Hash, _Pred, _Alloc >::operator= (
            initializer_list< value_type > __l)  [inline]
```

Unordered_set list assignment operator.

Parameters

↩ _↩ ↩ _↩ *l*	An initializer_list.

This function fills an unordered_set with copies of the elements in the initializer list *__l*.
Note that the assignment completely changes the unordered_set and that the resulting unordered_set's size is the same as the number of elements assigned.

operator=() [3/3]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
unordered_set & std::unordered_set< _Value, _Hash, _Pred, _Alloc >::operator= (
            unordered_set< _Value, _Hash, _Pred, _Alloc > && )  [default]
```

Move assignment operator.

rehash()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::rehash (
            size_type __n)  [inline]
```

May rehash the unordered_set.

Parameters

_↩ *_n*	The new number of buckets.

Rehash will occur only if the new number of buckets respect the unordered_set maximum load factor.

reserve()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::reserve (
            size_type __n)  [inline]
```

Prepare the unordered_set for a specified number of elements.

Parameters

_↩ *_n*	Number of elements required.

Same as rehash(ceil(n / max_load_factor())).

size()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::size () const  [inline], [noexcept]
```

Returns the size of the unordered_set.

swap()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::swap (
            unordered_set< _Value, _Hash, _Pred, _Alloc > & __x)  [inline], [noexcept]
```

Swaps data with another unordered_set.

Parameters

_↩ _x	An unordered_set of the same element and allocator types.

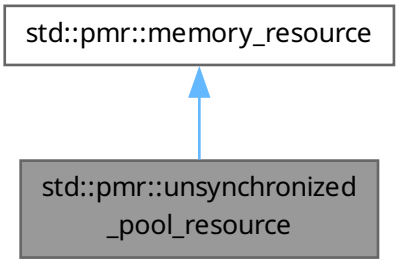
This exchanges the elements between two sets in constant time. Note that the global std::swap() function is specialized such that std::swap(s1,s2) will feed to this function.
The documentation for this class was generated from the following file:

- unordered_set.h

5.1003 std::pmr::unsynchronized_pool_resource Class Reference

```
#include <memory_resource>
```

Inheritance diagram for std::pmr::unsynchronized_pool_resource:



Public Member Functions

- **unsynchronized_pool_resource** (const pool_options &__opts)
- **unsynchronized_pool_resource** (const pool_options &__opts, memory_resource *__upstream)
- **unsynchronized_pool_resource** (const unsynchronized_pool_resource &)=delete
- **unsynchronized_pool_resource** (memory_resource *__upstream)
- void * **allocate** (size_t __bytes, size_t __alignment=_S_max_align)
- void **deallocate** (void *__p, size_t __bytes, size_t __alignment=_S_max_align)
- bool **is_equal** (const memory_resource &__other) const noexcept
- unsynchronized_pool_resource & **operator=** (const unsynchronized_pool_resource &)=delete
- pool_options **options** () const noexcept
- void **release** ()
- memory_resource * **upstream_resource** () const noexcept

Protected Member Functions

- void * do_allocate (size_t __bytes, size_t __alignment) override
- void do_deallocate (void *__p, size_t __bytes, size_t __alignment) override
- bool do_is_equal (const memory_resource &__other) const noexcept override

5.1003.1 Detailed Description

A non-thread-safe memory resource that manages pools of fixed-size blocks.

Since

C++17

5.1003.2 Member Function Documentation

do_allocate()

```
void * std::pmr::unsynchronized_pool_resource::do_allocate (
            size_t __bytes,
            size_t __alignment)  [override], [protected], [virtual]
```

Implements std::pmr::memory_resource.

do_deallocate()

```
void std::pmr::unsynchronized_pool_resource::do_deallocate (
            void * __p,
            size_t __bytes,
            size_t __alignment)  [override], [protected], [virtual]
```

Implements std::pmr::memory_resource.

do_is_equal()

```
bool std::pmr::unsynchronized_pool_resource::do_is_equal (
            const memory_resource & __other) const  [inline], [override], [protected], [virtual],
[noexcept]
```

Implements std::pmr::memory_resource.

The documentation for this class was generated from the following file:

- memory_resource

5.1004 std::uses_allocator< typename, typename > Struct Template Reference

5.1004.1 Detailed Description

template<typename, typename>
struct std::uses_allocator< typename, typename >

Declare uses_allocator so it can be specialized in <queue> etc.
The documentation for this struct was generated from the following file:

- memoryfwd.h

5.1005 std::uses_allocator< tuple< _Types... >, _Alloc > Struct Template Reference

`#include <tuple>`

5.1005.1 Detailed Description

template<typename... _Types, typename _Alloc>
struct std::uses_allocator< tuple< _Types... >, _Alloc >

Partial specialization for tuples.
The documentation for this struct was generated from the following file:

- tuple

5.1006 std::chrono::utc_clock Class Reference

`#include <chrono>`

Public Types

- using **duration**
- using **period**
- using **rep**
- using **time_point**

Static Public Member Functions

- template<typename _Duration>
  static utc_time< common_type_t< _Duration, seconds > > **from_sys** (const sys_time< _Duration > &__t)
- static time_point **now** ()
- template<typename _Duration>
  static sys_time< common_type_t< _Duration, seconds > > **to_sys** (const utc_time< _Duration > &__t)

Static Public Attributes

- static constexpr bool **is_steady**

5.1006.1 Detailed Description

A clock that measures Universal Coordinated Time (UTC).
The epoch is 1970-01-01 00:00:00.

Since

C++20

The documentation for this class was generated from the following file:

- chrono

5.1007 std::valarray< _Tp > Class Template Reference

`#include <valarray>`

Public Types

- typedef _Tp **value_type**

Public Member Functions

- valarray () noexcept
- template<class _Dom>
  valarray (const _Expr< _Dom, _Tp > &__e)
- valarray (const _Tp &, size_t)
- template<typename _Tp>
  valarray (const _Tp *__restrict__ __p, size_t __n)
- valarray (const _Tp *__restrict__, size_t)
- valarray (const gslice_array< _Tp > &)
- valarray (const indirect_array< _Tp > &)
- valarray (const mask_array< _Tp > &)
- valarray (const slice_array< _Tp > &)
- valarray (const valarray &)
- valarray (initializer_list< _Tp >)
- valarray (size_t)
- valarray (valarray &&) noexcept
- _Expr< _ValFunClos< _ValArray, _Tp >, _Tp > apply (_Tp __func(_Tp)) const
- _Expr< _RefFunClos< _ValArray, _Tp >, _Tp > apply (_Tp __func(const _Tp &)) const
- valarray< _Tp > cshift (int __n) const
- _Tp max () const
- _Tp min () const
- _UnaryOp< __logical_not >::_Rt operator! () const
- template<class _Dom>
  valarray< _Tp > & **operator%=** (const _Expr< _Dom, _Tp > &)
- valarray< _Tp > & operator%= (const _Tp &)
- valarray< _Tp > & operator%= (const valarray< _Tp > &)
- template<class _Dom>
  valarray< _Tp > & **operator&=** (const _Expr< _Dom, _Tp > &)
- valarray< _Tp > & operator&= (const _Tp &)
- valarray< _Tp > & operator&= (const valarray< _Tp > &)
- template<class _Dom>
  valarray< _Tp > & **operator*=** (const _Expr< _Dom, _Tp > &)
- valarray< _Tp > & operator*= (const _Tp &)
- valarray< _Tp > & operator*= (const valarray< _Tp > &)
- _UnaryOp< __unary_plus >::_Rt operator+ () const
- template<class _Dom>
  valarray< _Tp > & **operator+=** (const _Expr< _Dom, _Tp > &)
- valarray< _Tp > & operator+= (const _Tp &)
- valarray< _Tp > & operator+= (const valarray< _Tp > &)
- _UnaryOp< __negate >::_Rt operator- () const
- template<class _Dom>
  valarray< _Tp > & **operator-=** (const _Expr< _Dom, _Tp > &)
- valarray< _Tp > & operator-= (const _Tp &)

- valarray< _Tp > & operator-= (const valarray< _Tp > &)
- template<class _Dom>
  valarray< _Tp > & **operator/=** (const _Expr< _Dom, _Tp > &)
- valarray< _Tp > & operator/= (const _Tp &)
- valarray< _Tp > & operator/= (const valarray< _Tp > &)
- template<class _Dom>
  valarray< _Tp > & **operator**<<**=** (const _Expr< _Dom, _Tp > &)
- valarray< _Tp > & operator<<= (const _Tp &)
- valarray< _Tp > & operator<<= (const valarray< _Tp > &)
- template<class _Dom>
  valarray< _Tp > & **operator=** (const _Expr< _Dom, _Tp > &)
- valarray< _Tp > & operator= (const _Tp &__t)
- valarray< _Tp > & operator= (const gslice_array< _Tp > &__ga)
- valarray< _Tp > & operator= (const indirect_array< _Tp > &__ia)
- valarray< _Tp > & operator= (const mask_array< _Tp > &__ma)
- valarray< _Tp > & operator= (const slice_array< _Tp > &__sa)
- valarray< _Tp > & operator= (const valarray< _Tp > &__v)
- valarray & operator= (initializer_list< _Tp > __l)
- valarray< _Tp > & operator= (valarray< _Tp > &&__v) noexcept
- template<class _Dom>
  valarray< _Tp > & **operator**>>**=** (const _Expr< _Dom, _Tp > &)
- valarray< _Tp > & operator>>= (const _Tp &)
- valarray< _Tp > & operator>>= (const valarray< _Tp > &)
- gslice_array< _Tp > operator[ ] (const gslice &__s)
- _Expr< _GClos< _ValArray, _Tp >, _Tp > operator[ ] (const gslice &__s) const
- mask_array< _Tp > operator[ ] (const valarray< bool > &__m)
- valarray< _Tp > operator[ ] (const valarray< bool > &__m) const
- indirect_array< _Tp > operator[ ] (const valarray< size_t > &__i)
- _Expr< _IClos< _ValArray, _Tp >, _Tp > operator[ ] (const valarray< size_t > &__i) const
- _Tp & operator[ ] (size_t __i) noexcept
- const _Tp & **operator[ ]** (size_t) const noexcept
- slice_array< _Tp > operator[ ] (slice __s)
- _Expr< _SClos< _ValArray, _Tp >, _Tp > operator[ ] (slice __s) const
- template<class _Dom>
  valarray< _Tp > & **operator**^**=** (const _Expr< _Dom, _Tp > &)
- valarray< _Tp > & operator^= (const _Tp &)
- valarray< _Tp > & operator^= (const valarray< _Tp > &)
- template<class _Dom>
  valarray< _Tp > & **operator**|**=** (const _Expr< _Dom, _Tp > &)
- valarray< _Tp > & operator|= (const _Tp &)
- valarray< _Tp > & operator|= (const valarray< _Tp > &)
- _UnaryOp< __bitwise_not >::_Rt operator~ () const
- void resize (size_t __size, _Tp __c=_Tp())
- valarray< _Tp > shift (int __n) const
- size_t size () const
- _Tp sum () const
- void swap (valarray< _Tp > &__v) noexcept

Friends

- struct **_Array**< **_Tp** >

5.1007.1 Detailed Description

template<class _Tp>
class std::valarray< _Tp >

Smart array designed to support numeric processing.
A valarray is an array that provides constraints intended to allow for effective optimization of numeric array processing by reducing the aliasing that can result from pointer representations. It represents a one-dimensional array from which different multidimensional subsets can be accessed and modified.

Template Parameters

_Tp	Type of object in the array.

5.1007.2 Constructor & Destructor Documentation

valarray()

```
template<class _Tp>
std::valarray< _Tp >::valarray (
            const _Tp * __restrict__,
            size_t )
```

Construct an array initialized to the first *n* elements of *t*.

5.1007.3 Member Function Documentation

operator"!()

```
template<class _Tp>
_UnaryOp< __logical_not >::_Rt std::valarray< _Tp >::operator!  () const
```

Return a new valarray by applying unary ! to each element.

operator%=() [1/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator%= (
            const _Tp & )
```

Set each element e of array to e % *t*.

operator%=() [2/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator%= (
            const valarray< _Tp > & )
```

Modulo elements of array by corresponding elements of *v*.

operator&=() [1/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator&= (
            const _Tp & )
```

Set each element e of array to e & *t*.

operator&=() [2/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator&= (
            const valarray< _Tp > & )
```

Logical and corresponding elements of *v* with elements of array.

operator*=() [1/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator*= (
            const _Tp & )
```

Multiply each element of array by *t*.

operator*=() [2/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator*= (
            const valarray< _Tp > & )
```

Multiply elements of array by corresponding elements of *v*.

operator+()

```
template<class _Tp>
_UnaryOp< __unary_plus >::_Rt std::valarray< _Tp >::operator+ () const
```

Return a new valarray by applying unary + to each element.

operator+=() [1/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator+= (
            const _Tp & )
```

Add *t* to each element of array.

operator+=() [2/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator+= (
            const valarray< _Tp > & )
```

Add corresponding elements of *v* to elements of array.

operator-()

```
template<class _Tp>
_UnaryOp< __negate >::_Rt std::valarray< _Tp >::operator- () const
```

Return a new valarray by applying unary - to each element.

operator-=() [1/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator-= (
            const _Tp & )
```

Subtract *t* to each element of array.

operator-=() [2/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator-= (
            const valarray< _Tp > & )
```

Subtract corresponding elements of *v* from elements of array.

operator/=() [1/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator/= (
            const _Tp & )
```

Divide each element of array by *t*.

operator/=() [2/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator/= (
            const valarray< _Tp > & )
```

Divide elements of array by corresponding elements of *v*.

operator<<=() [1/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator<<= (
            const _Tp & )
```

Left shift each element e of array by *t* bits.

operator<<=() [2/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator<<= (
            const valarray< _Tp > & )
```

Left shift elements of array by corresponding elements of *v*.

operator>>=() [1/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator>>= (
            const _Tp & )
```

Right shift each element e of array by *t* bits.

operator>>=() [2/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator>>= (
            const valarray< _Tp > & )
```

Right shift elements of array by corresponding elements of *v*.

operator^=() [1/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator^= (
            const _Tp & )
```

Set each element e of array to e ^ *t*.

operator^=() [2/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator^= (
            const valarray< _Tp > & )
```

Logical xor corresponding elements of *v* with elements of array.

operator"|=() [1/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator|= (
            const _Tp & )
```

Set each element e of array to e | *t*.

operator"|=() [2/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator|= (
            const valarray< _Tp > & )
```

Logical or corresponding elements of *v* with elements of array.

operator~()

```
template<class _Tp>
_UnaryOp< __bitwise_not >::_Rt std::valarray< _Tp >::operator~ () const
```

Return a new valarray by applying unary ~ to each element.

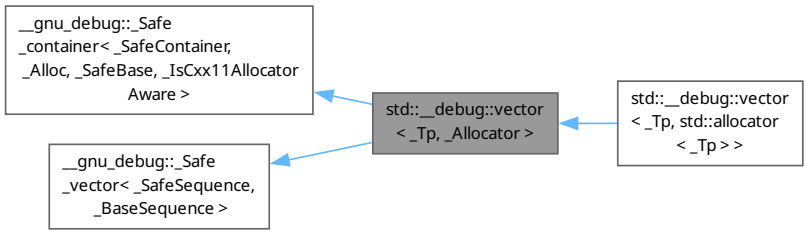
The documentation for this class was generated from the following file:

- valarray

5.1008 std::__debug::vector< _Tp, _Allocator > Class Template Reference

```
#include <vector>
```

Inheritance diagram for std::__debug::vector< _Tp, _Allocator >:



Public Types

- typedef _Allocator **allocator_type**
- typedef __gnu_debug::_Safe_iterator< _Base_const_iterator, vector > **const_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**
- typedef std::reverse_iterator< const_iterator > **const_reverse_iterator**

- typedef _Base::difference_type **difference_type**
- typedef __gnu_debug::_Safe_iterator< _Base_iterator, vector > **iterator**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef std::reverse_iterator< iterator > **reverse_iterator**
- typedef _Base::size_type **size_type**
- typedef _Tp **value_type**

Public Member Functions

- constexpr vector (_Base_ref __x)
- template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  constexpr **vector** (_InputIterator __first, _InputIterator __last, const _Allocator &__a=_Allocator())
- constexpr **vector** (const _Allocator &__a) noexcept
- **vector** (const vector &)=default
- constexpr **vector** (const vector &__x, const __type_identity_t< allocator_type > &__a)
- constexpr **vector** (initializer_list< value_type > __l, const allocator_type &__a=allocator_type())
- constexpr **vector** (size_type __n, const __type_identity_t< _Tp > &__value, const _Allocator &__a=_Allocator())
- constexpr **vector** (size_type __n, const _Allocator &__a=_Allocator())
- **vector** (vector &&)=default
- constexpr **vector** (vector &&__x, const __type_identity_t< allocator_type > &__a) noexcept(std::is_nothrow_constructible< _Base, _Base, const allocator_type & >::value)
- constexpr const _Base & **_M_base** () const noexcept
- constexpr _Base & **_M_base** () noexcept
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  constexpr void **assign** (_InputIterator __first, _InputIterator __last)
- constexpr void **assign** (initializer_list< value_type > __l)
- constexpr void **assign** (size_type __n, const _Tp &__u)
- constexpr const_reference **back** () const noexcept
- constexpr reference **back** () noexcept
- constexpr const_iterator **begin** () const noexcept
- constexpr iterator **begin** () noexcept
- constexpr size_type **capacity** () const noexcept
- constexpr const_iterator **cbegin** () const noexcept
- constexpr const_iterator **cend** () const noexcept
- constexpr void **clear** () noexcept
- constexpr const_reverse_iterator **crbegin** () const noexcept
- constexpr const_reverse_iterator **crend** () const noexcept
- template<typename... _Args>
  constexpr iterator **emplace** (const_iterator __position, _Args &&... __args)
- template<typename... _Args>
  constexpr reference **emplace_back** (_Args &&... __args)
- constexpr const_iterator **end** () const noexcept
- constexpr iterator **end** () noexcept
- constexpr iterator **erase** (const_iterator __first, const_iterator __last)
- constexpr iterator **erase** (const_iterator __position)
- constexpr const_reference **front** () const noexcept
- constexpr reference **front** () noexcept
- template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  constexpr iterator **insert** (const_iterator __position, _InputIterator __first, _InputIterator __last)

- template<typename _Up = _Tp>
  constexpr __gnu_cxx::__enable_if<!std::__are_same< _Up, bool >::__value, iterator >::__type **insert** (const_iterator __position, _Tp &&__x)
- constexpr iterator **insert** (const_iterator __position, const _Tp &__x)
- constexpr iterator **insert** (const_iterator __position, initializer_list< value_type > __l)
- constexpr iterator **insert** (const_iterator __position, size_type __n, const _Tp &__x)
- vector & **operator=** (const vector &)=default
- constexpr vector & **operator=** (initializer_list< value_type > __l)
- vector & **operator=** (vector &&)=default
- constexpr const_reference **operator[ ]** (size_type __n) const noexcept
- constexpr reference **operator[ ]** (size_type __n) noexcept
- constexpr void **pop_back** () noexcept
- template<typename _Up = _Tp>
  constexpr __gnu_cxx::__enable_if<!std::__are_same< _Up, bool >::__value, void >::__type **push_back** (_Tp &&__x)
- constexpr void **push_back** (const _Tp &__x)
- constexpr const_reverse_iterator **rbegin** () const noexcept
- constexpr reverse_iterator **rbegin** () noexcept
- constexpr const_reverse_iterator **rend** () const noexcept
- constexpr reverse_iterator **rend** () noexcept
- constexpr void **reserve** (size_type __n)
- constexpr void **resize** (size_type __sz)
- constexpr void **resize** (size_type __sz, const _Tp &__c)
- constexpr void **shrink_to_fit** ()
- constexpr void **swap** (vector &__x) noexcept(/*conditional */)

Protected Member Functions

- bool **_M_requires_reallocation** (size_type __elements) const noexcept
- constexpr void **_M_swap** (const _Safe_container &__x) const noexcept
- constexpr void **_M_update_guaranteed_capacity** () noexcept

Protected Attributes

- size_type **_M_guaranteed_capacity**

Friends

- template<typename _ItT, typename _SeqT, typename _CatT>
  class **::__gnu_debug::_Safe_iterator**

5.1008.1 Detailed Description

template<typename _Tp, typename _Allocator = std::allocator<_Tp>>
class std::__debug::vector< _Tp, _Allocator >

Class std::vector with safety/checking/debug instrumentation.

5.1008.2 Constructor & Destructor Documentation

vector()

```
template<typename _Tp, typename _Allocator = std::allocator<_Tp>>
std::__debug::vector< _Tp, _Allocator >::vector (
            _Base_ref __x)  [inline], [constexpr]
```

Construction from a normal-mode vector.

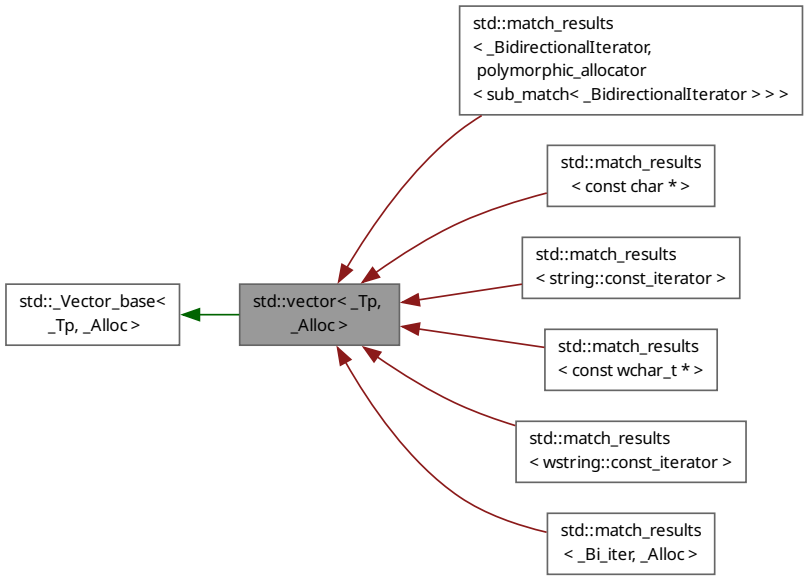
The documentation for this class was generated from the following file:

- debug/vector

5.1009 std::vector< _Tp, _Alloc > Class Template Reference

```
#include <vector>
```

Inheritance diagram for std::vector< _Tp, _Alloc >:



Public Types

- typedef _Alloc **allocator_type**
- typedef __gnu_cxx::__normal_iterator< const_pointer, vector > **const_iterator**
- typedef _Alloc_traits::const_pointer **const_pointer**
- typedef _Alloc_traits::const_reference **const_reference**
- typedef std::reverse_iterator< const_iterator > **const_reverse_iterator**
- typedef ptrdiff_t **difference_type**
- typedef __gnu_cxx::__normal_iterator< pointer, vector > **iterator**
- typedef _Base::pointer **pointer**
- typedef _Alloc_traits::reference **reference**
- typedef std::reverse_iterator< iterator > **reverse_iterator**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- vector ()=default
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  constexpr vector (_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type())
- constexpr vector (const allocator_type &__a) noexcept
- constexpr vector (const vector &__x)
- constexpr vector (const vector &__x, const __type_identity_t< allocator_type > &__a)
- constexpr vector (initializer_list< value_type > __l, const allocator_type &__a=allocator_type())
- constexpr vector (size_type __n, const allocator_type &__a=allocator_type())
- constexpr vector (size_type __n, const value_type &__value, const allocator_type &__a=allocator_type())
- vector (vector &&) noexcept=default
- constexpr vector (vector &&__rv, const __type_identity_t< allocator_type > &__m) noexcept(noexcept(vector(std::declval< vector && >(), std::declval< const allocator_type & >(), std::declval< typename _Alloc_traits::is_always_equal >())))
- constexpr ~vector () noexcept
- template<typename... _Args>
  constexpr auto **_M_emplace_aux** (const_iterator __position, _Args &&... __args) -> iterator
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  constexpr void assign (_InputIterator __first, _InputIterator __last)
- constexpr void assign (initializer_list< value_type > __l)
- constexpr void assign (size_type __n, const value_type &__val)
- constexpr reference at (size_type __n)
- constexpr const_reference at (size_type __n) const
- constexpr const_reference back () const noexcept
- constexpr reference back () noexcept
- constexpr const_iterator begin () const noexcept
- constexpr iterator begin () noexcept
- constexpr size_type capacity () const noexcept
- constexpr const_iterator cbegin () const noexcept
- constexpr const_iterator cend () const noexcept
- constexpr void clear () noexcept
- constexpr const_reverse_iterator crbegin () const noexcept
- constexpr const_reverse_iterator crend () const noexcept
- constexpr const _Tp * **data** () const noexcept
- constexpr _Tp * data () noexcept
- template<typename... _Args>
  constexpr iterator emplace (const_iterator __position, _Args &&... __args)
- template<typename... _Args>
  constexpr reference **emplace_back** (_Args &&... __args)
- constexpr bool empty () const noexcept
- constexpr const_iterator end () const noexcept
- constexpr iterator end () noexcept
- constexpr iterator erase (const_iterator __first, const_iterator __last)
- constexpr iterator erase (const_iterator __position)
- constexpr const_reference front () const noexcept
- constexpr reference front () noexcept
- constexpr allocator_type get_allocator () const noexcept
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  constexpr iterator insert (const_iterator __position, _InputIterator __first, _InputIterator __last)
- constexpr iterator insert (const_iterator __position, const value_type &__x)
- constexpr iterator insert (const_iterator __position, initializer_list< value_type > __l)

- constexpr iterator insert (const_iterator __position, size_type __n, const value_type &__x)
- constexpr iterator insert (const_iterator __position, value_type &&__x)
- constexpr size_type max_size () const noexcept
- constexpr vector & operator= (const vector &__x)
- constexpr vector & operator= (initializer_list< value_type > __l)
- constexpr vector & operator= (vector &&__x) noexcept(_Alloc_traits::_S_nothrow_move())
- constexpr const_reference operator[ ] (size_type __n) const noexcept
- constexpr reference operator[ ] (size_type __n) noexcept
- constexpr void pop_back () noexcept
- constexpr void push_back (const value_type &__x)
- constexpr void **push_back** (value_type &&__x)
- constexpr const_reverse_iterator rbegin () const noexcept
- constexpr reverse_iterator rbegin () noexcept
- constexpr const_reverse_iterator rend () const noexcept
- constexpr reverse_iterator rend () noexcept
- constexpr void reserve (size_type __n)
- constexpr void resize (size_type __new_size)
- constexpr void resize (size_type __new_size, const value_type &__x)
- constexpr void shrink_to_fit ()
- constexpr size_type size () const noexcept
- constexpr void swap (vector &__x) noexcept

Protected Member Functions

- constexpr pointer **_M_allocate** (size_t __n)
- template<typename _ForwardIterator>
  constexpr pointer _M_allocate_and_copy (size_type __n, _ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator>
  constexpr void **_M_assign_aux** (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- template<typename _InputIterator>
  constexpr void **_M_assign_aux** (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- template<typename _InputIterator>
  constexpr void **_M_assign_dispatch** (_InputIterator __first, _InputIterator __last, __false_type)
- template<typename _Integer>
  constexpr void **_M_assign_dispatch** (_Integer __n, _Integer __val, __true_type)
- constexpr size_type **_M_check_len** (size_type __n, const char *__s) const
- constexpr void **_M_create_storage** (size_t __n)
- constexpr void **_M_deallocate** (pointer __p, size_t __n)
- constexpr void **_M_default_append** (size_type __n)
- constexpr void **_M_default_initialize** (size_type __n)
- template<typename... _Args>
  constexpr iterator **_M_emplace_aux** (const_iterator __position, _Args &&... __args)
- constexpr iterator **_M_emplace_aux** (const_iterator __position, value_type &&__v)
- constexpr iterator **_M_erase** (iterator __first, iterator __last)
- constexpr iterator **_M_erase** (iterator __position)
- constexpr void **_M_erase_at_end** (pointer __pos) noexcept
- constexpr void **_M_fill_append** (size_type __n, const value_type &__x)
- constexpr void **_M_fill_assign** (size_type __n, const value_type &__val)
- constexpr void **_M_fill_initialize** (size_type __n, const value_type &__value)
- constexpr void **_M_fill_insert** (iterator __pos, size_type __n, const value_type &__x)
- constexpr const _Tp_alloc_type & **_M_get_Tp_allocator** () const noexcept

- constexpr _Tp_alloc_type & **_M_get_Tp_allocator** () noexcept
- template<typename _Arg>
  constexpr void **_M_insert_aux** (iterator __position, _Arg &&__arg)
- template<typename _InputIterator>
  constexpr void **_M_insert_dispatch** (iterator __pos, _InputIterator __first, _InputIterator __last, __false_type)
- template<typename _Integer>
  constexpr void **_M_insert_dispatch** (iterator __pos, _Integer __n, _Integer __val, __true_type)
- constexpr iterator **_M_insert_rval** (const_iterator __position, value_type &&__v)
- constexpr void _M_range_check (size_type __n) const
- template<typename _ForwardIterator>
  constexpr void **_M_range_initialize** (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- template<typename _InputIterator>
  constexpr void **_M_range_initialize** (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- template<typename _Iterator, typename _Sentinel>
  constexpr void **_M_range_initialize_n** (_Iterator __first, _Sentinel __last, size_type __n)
- template<typename _ForwardIterator>
  constexpr void **_M_range_insert** (iterator __pos, _ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- template<typename _InputIterator>
  constexpr void **_M_range_insert** (iterator __pos, _InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- template<typename... _Args>
  constexpr void **_M_realloc_append** (_Args &&... __args)
- template<typename... _Args>
  constexpr void **_M_realloc_insert** (iterator __position, _Args &&... __args)
- constexpr bool **_M_shrink_to_fit** ()

Static Protected Member Functions

- static constexpr size_type **_S_check_init_len** (size_type __n, const allocator_type &__a)
- static constexpr size_type **_S_max_size** (const _Tp_alloc_type &__a) noexcept

Protected Attributes

- _Vector_impl **_M_impl**

5.1009.1 Detailed Description

template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
class std::vector< _Tp, _Alloc >

A standard container which offers fixed time access to individual elements in any order.

Since

C++98

Template Parameters

_Tp	Type of element.
_Alloc	Allocator type, defaults to allocator<_Tp>.

Meets the requirements of a `container`, a `reversible container`, and a `sequence`, including the `optional sequence requirements` with the exception of `push_front` and `pop_front`.
In some terminology a vector can be described as a dynamic C-style array, it offers fast and efficient access to individual elements in any order and saves the user from worrying about memory and size allocation. Subscripting ([]) access is also provided as with C-style arrays.

5.1009.2 Constructor & Destructor Documentation

vector() [1/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::vector ()  [default]
```

Creates a vector with no elements.
Referenced by insert(), and operator=().

vector() [2/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::vector (
            const allocator_type & __a)  [inline], [explicit], [constexpr], [noexcept]
```

Creates a vector with no elements.

Parameters

_↩ _a	An allocator object.

vector() [3/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::vector (
            size_type __n,
            const allocator_type & __a = allocator_type())  [inline], [explicit], [constexpr]
```

Creates a vector with default constructed elements.

Parameters

_↩ _n	The number of elements to initially create.
_↩ _a	An allocator.

This constructor fills the vector with *__n* default constructed elements.

vector() [4/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::vector (
            size_type __n,
            const value_type & __value,
            const allocator_type & __a = allocator_type())  [inline], [constexpr]
```

Creates a vector with copies of an exemplar element.

Parameters

__n	The number of elements to initially create.
__value	An element to copy.
__a	An allocator.

This constructor fills the vector with *__n* copies of *__value*.

vector() [5/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::vector (
            const vector< _Tp, _Alloc > & __x)  [inline], [constexpr]
```

Vector copy constructor.

Parameters

_↩ _x	A vector of identical element and allocator types.

All the elements of __x are copied, but any unused capacity in __x will not be copied (i.e. capacity() == size() in the new vector).
The newly-created vector uses a copy of the allocator object used by __x (unless the allocator traits dictate a different object).

vector() [6/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::vector (
            vector< _Tp, _Alloc > && )  [default], [noexcept]
```

Vector move constructor.
The newly-created vector contains the exact contents of the moved instance. The contents of the moved instance are a valid, but unspecified vector.

vector() [7/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::vector (
            const vector< _Tp, _Alloc > & __x,
            const __type_identity_t< allocator_type > & __a)  [inline], [constexpr]
```

Copy constructor with alternative allocator.

vector() [8/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::vector (
            vector< _Tp, _Alloc > && __rv,
            const __type_identity_t< allocator_type > & __m)  [inline], [constexpr], [noexcept]
```

Move constructor with alternative allocator.

vector() [9/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::vector (
            initializer_list< value_type > __l,
            const allocator_type & __a = allocator_type())  [inline], [constexpr]
```

Builds a vector from an initializer list.

Parameters

_↩ _l	An initializer_list.
_↩ _a	An allocator.

Create a vector consisting of copies of the elements in the initializer_list __*l*.
This will call the element type's copy constructor N times (where N is __*l.size()*) and do no memory reallocation.

vector() [10/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
std::vector< _Tp, _Alloc >::vector (
            _InputIterator __first,
            _InputIterator __last,
            const allocator_type & __a = allocator_type())  [inline], [constexpr]
```

Builds a vector from a range.

Parameters

__*first*	An input iterator.
__*last*	An input iterator.
__*a*	An allocator.

Create a vector consisting of copies of the elements from [first,last).
If the iterators are forward, bidirectional, or random-access, then this will call the elements' copy constructor N times (where N is distance(first,last)) and do no memory reallocation. But if only input iterators are used, then this will do at most 2N calls to the copy constructor, and logN memory reallocations.

~vector()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::~vector ()  [inline], [constexpr], [noexcept]
```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

5.1009.3 Member Function Documentation

_M_allocate_and_copy()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _ForwardIterator>
pointer std::vector< _Tp, _Alloc >::_M_allocate_and_copy (
            size_type __n,
            _ForwardIterator __first,
            _ForwardIterator __last)  [inline], [constexpr], [protected]
```

Memory expansion handler. Uses the member allocation function to obtain *n* bytes of memory, and then copies [first,last) into it.
Referenced by operator=(), and reserve().

_M_range_check()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::_M_range_check (
            size_type __n) const  [inline], [constexpr], [protected]
```

Safety check used only from at().
Referenced by std::vector< _Tp, polymorphic_allocator< _Tp > >::at(), and std::vector< _Tp, polymorphic_allocator< _Tp > >::at().

assign() [1/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
void std::vector< _Tp, _Alloc >::assign (
            _InputIterator __first,
            _InputIterator __last)  [inline], [constexpr]
```

Assigns a range to a vector.

Parameters

__first	An input iterator.
__last	An input iterator.

This function fills a vector with copies of the elements in the range [__first,__last).
Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned.

assign() [2/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::assign (
            initializer_list< value_type > __l)  [inline], [constexpr]
```

Assigns an initializer list to a vector.

Parameters

↩ _↩ ↩ _↩ *l*	An initializer_list.

This function fills a vector with copies of the elements in the initializer list *__l*.
Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned.

assign() [3/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::assign (
            size_type __n,
            const value_type & __val)  [inline], [constexpr]
```

Assigns a given value to a vector.

Parameters

__n	Number of elements to be assigned.
__val	Value to be assigned.

This function fills a vector with *__n* copies of the given value. Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned.

at() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::vector< _Tp, _Alloc >::at (
            size_type __n)  [inline], [nodiscard], [constexpr]
```

Provides access to the data contained in the vector.

Parameters

_↩ _n	The index of the element for which data should be accessed.

Returns

Read/write reference to data.

Exceptions

std::out_of_range	If *__n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the vector. The function throws out_of_range if the check fails.

at() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::vector< _Tp, _Alloc >::at (
            size_type __n) const  [inline], [nodiscard], [constexpr]
```

Provides access to the data contained in the vector.

Parameters

_↩ _n	The index of the element for which data should be accessed.

Returns

Read-only (constant) reference to data.

Exceptions

std::out_of_range	If *__n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the vector. The function throws out_of_range if the check fails.

back() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::vector< _Tp, _Alloc >::back () const  [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read-only (constant) reference to the data at the last element of the vector.

back() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::vector< _Tp, _Alloc >::back ()  [inline], [nodiscard], [constexpr], [noexcept]
```

Returns a read/write reference to the data at the last element of the vector.

begin() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::vector< _Tp, _Alloc >::begin () const  [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the vector. Iteration is done in ordinary element order.

begin() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector< _Tp, _Alloc >::begin ()  [inline], [nodiscard], [constexpr], [noexcept]
```

Returns a read/write iterator that points to the first element in the vector. Iteration is done in ordinary element order.
Referenced by std::vector< _Tp, polymorphic_allocator< _Tp > >::vector(), std::vector< _Tp, polymorphic_allocator< _Tp > >::crend(), std::vector< _Tp, polymorphic_allocator< _Tp > >::empty(), std::vector< _Tp, polymorphic_allocator< _Tp > >::erase(), std::vector< _Tp, polymorphic_allocator< _Tp > >::erase(), std::vector< _Tp, polymorphic_allocator< _Tp > >::front(), std::vector< _Tp, polymorphic_allocator< _Tp > >::front(), std::vector< _Tp, polymorphic_allocator< _Tp > >::insert(), std::vector< _Tp, polymorphic_allocator< _Tp > >::insert(), std::vector< _Tp, polymorphic_allocator< _Tp > >::insert(), __gnu_parallel::multiway_merge_exact_splitting(), std::operator<=>(), operator=(), std::operator==(), std::vector< _Tp, polymorphic_allo and std::vector< _Tp, polymorphic_allocator< _Tp > >::rend().

capacity()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::vector< _Tp, _Alloc >::capacity () const  [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns the total number of elements that the vector can hold before needing to allocate more memory.
Referenced by operator=(), and reserve().

cbegin()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::vector< _Tp, _Alloc >::cbegin () const  [inline], [constexpr], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the vector. Iteration is done in ordinary element order.
Referenced by std::vector< _Tp, polymorphic_allocator< _Tp > >::erase(), std::vector< _Tp, polymorphic_allocator< _Tp > >::erase(), std::vector< _Tp, polymorphic_allocator< _Tp > >::insert(), std::vector< _Tp, polymorphic_allocator< _Tp > >::insert(), and std::vector< _Tp, polymorphic_allocator< _Tp > >::insert().

cend()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::vector< _Tp, _Alloc >::cend () const  [inline], [constexpr], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

clear()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::clear ()  [inline], [constexpr], [noexcept]
```

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.
Referenced by operator=().

crbegin()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::vector< _Tp, _Alloc >::crbegin () const  [inline], [constexpr], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

crend()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::vector< _Tp, _Alloc >::crend () const  [inline], [constexpr], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

data()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
_Tp * std::vector< _Tp, _Alloc >::data ()  [inline], [nodiscard], [constexpr], [noexcept]
```

Returns a pointer such that [data(), data() + size()) is a valid range. For a non-empty vector, data() == &front().

emplace()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename...  _Args>
iterator std::vector< _Tp, _Alloc >::emplace (
            const_iterator __position,
            _Args &&...  __args)  [inline], [constexpr]
```

Inserts an object in vector before specified iterator.

Parameters

__position	A const_iterator into the vector.
__args	Arguments.

Returns

An iterator that points to the inserted data.

This function will insert an object of type T constructed with T(std::forward<Args>(args)...) before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using std::list.

empty()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
bool std::vector< _Tp, _Alloc >::empty () const  [inline], [nodiscard], [constexpr], [noexcept]
```

Returns true if the vector is empty. (Thus begin() would equal end().)
Referenced by std::piecewise_constant_distribution< _RealType >::densities(), std::piecewise_linear_distribution< _RealType >::densiti
and std::discrete_distribution< _IntType >::probabilities().

end() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::vector< _Tp, _Alloc >::end () const  [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

end() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector< _Tp, _Alloc >::end ()  [inline], [nodiscard], [constexpr], [noexcept]
```

Returns a read/write iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

Referenced by std::vector< _Tp, polymorphic_allocator< _Tp > >::vector(), std::vector< _Tp, polymorphic_allocator< _Tp > >::back(), std::vector< _Tp, polymorphic_allocator< _Tp > >::back(), std::vector< _Tp, polymorphic_allocator< _Tp > >::crbegin(), std::vector< _Tp, polymorphic_allocator< _Tp > >::empty(), __gnu_parallel::multiway_merge_exact_splitting(), std::operator<=>(), operator=(), std::operator==(), std::vector< _Tp, polymorphic_allocator< _Tp > >::rbegin(), and std::vector< _Tp, polymorphic_allocator< _Tp > >::rbegin().

erase() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector< _Tp, _Alloc >::erase (
            const_iterator __first,
            const_iterator __last)  [inline], [constexpr]
```

Remove a range of elements.

Parameters

__first	Iterator pointing to the first element to be erased.
__last	Iterator pointing to one past the last element to be erased.

Returns

An iterator pointing to the element pointed to by *__last* prior to erasing (or end()).

This function will erase the elements in the range [__first,__last) and shorten the vector accordingly.

Note This operation could be expensive and if it is frequently used the user should consider using std::list. The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector< _Tp, _Alloc >::erase (
            const_iterator __position)  [inline], [constexpr]
```

Remove element at given position.

Parameters

__position	Iterator pointing to element to be erased.

Returns

An iterator pointing to the next element (or end()).

This function will erase the element at the given position and thus shorten the vector by one.
Note This operation could be expensive and if it is frequently used the user should consider using std::list. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

front() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::vector< _Tp, _Alloc >::front () const  [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read-only (constant) reference to the data at the first element of the vector.

front() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::vector< _Tp, _Alloc >::front ()  [inline], [nodiscard], [constexpr], [noexcept]
```

Returns a read/write reference to the data at the first element of the vector.

get_allocator()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
allocator_type std::_Vector_base< _Tp, _Alloc >::get_allocator () const  [inline], [constexpr],
[noexcept]
```

Get a copy of the memory allocation object.
References get_allocator().
Referenced by get_allocator().

insert() [1/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
iterator std::vector< _Tp, _Alloc >::insert (
            const_iterator __position,
            _InputIterator __first,
            _InputIterator __last)  [inline], [constexpr]
```

Inserts a range into the vector.

Parameters

__position	A const_iterator into the vector.
__first	An input iterator.
__last	An input iterator.

Returns

An iterator that points to the inserted data.

This function will insert copies of the data in the range [__first,__last) into the vector before the location specified by *pos*. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using std::list.

insert() [2/5]

```
template<typename _Tp, typename _Alloc>
vector< _Tp, _Alloc >::iterator vector::insert (
            const_iterator __position,
            const value_type & __x)  [constexpr]
```

Inserts given value into vector before specified iterator.

Parameters

__position	A const_iterator into the vector.
__x	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using std::list.
References vector(), std::begin(), std::cbegin(), std::end(), insert(), and std::move().
Referenced by insert().

insert() [3/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector< _Tp, _Alloc >::insert (
            const_iterator __position,
            initializer_list< value_type > __l)  [inline], [constexpr]
```

Inserts an initializer_list into the vector.

Parameters

__position	An iterator into the vector.
__l	An initializer_list.

This function will insert copies of the data in the initializer_list *l* into the vector before the location specified by *position*. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using std::list.

insert() [4/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector< _Tp, _Alloc >::insert (
            const_iterator __position,
            size_type __n,
            const value_type & __x)  [inline], [constexpr]
```

Inserts a number of copies of given data into the vector.

Parameters

__position	A const_iterator into the vector.
__n	Number of elements to be inserted.
__x	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a specified number of copies of the given data before the location specified by *position*.
Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using std::list.

insert() [5/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector< _Tp, _Alloc >::insert (
            const_iterator __position,
            value_type && __x)  [inline], [constexpr]
```

Inserts given rvalue into vector before specified iterator.

Parameters

__position	A const_iterator into the vector.
__x	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using std::list.

max_size()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::vector< _Tp, _Alloc >::max_size () const  [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns the size() of the largest possible vector.
Referenced by reserve().

operator=() [1/3]

```
template<typename _Tp, typename _Alloc>
vector< _Tp, _Alloc > & vector::operator= (
            const vector< _Tp, _Alloc > & __x)  [constexpr]
```

Vector assignment operator.

Parameters

_↩ _x	A vector of identical element and allocator types.

All the elements of *__x* are copied, but any unused capacity in *__x* will not be copied.
Whether the allocator is copied depends on the allocator traits.
References vector(), std::__addressof(), std::_Destroy(), _M_allocate_and_copy(), begin(), capacity(), clear(), std::end(), end(), std::size(), and size().

operator=() [2/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
vector & std::vector< _Tp, _Alloc >::operator= (
            initializer_list< value_type > __l)  [inline], [constexpr]
```

Vector list assignment operator.

Parameters

↩ _↩ ↩ _↩ *l*	An initializer_list.

This function fills a vector with copies of the elements in the initializer list *__l*.
Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned.

operator=() [3/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
vector & std::vector< _Tp, _Alloc >::operator= (
            vector< _Tp, _Alloc > && __x)  [inline], [constexpr], [noexcept]
```

Vector move assignment operator.

Parameters

_↩ *_x*	A vector of identical element and allocator types.

The contents of *__x* are moved into this vector (without copying, if the allocators permit it). Afterwards *__x* is a valid, but unspecified vector.
Whether the allocator is moved depends on the allocator traits.

operator[]() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::vector< _Tp, _Alloc >::operator[] (
            size_type __n) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Subscript access to the data contained in the vector.

Parameters

_↩ *_n*	The index of the element for which data should be accessed.

Returns

Read-only (constant) reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out_of_range lookups are not defined. (For checked lookups see at().)

operator[]() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::vector< _Tp, _Alloc >::operator[] (
            size_type __n)  [inline], [nodiscard], [constexpr], [noexcept]
```

Subscript access to the data contained in the vector.

Parameters

_↩ _n	The index of the element for which data should be accessed.

Returns

Read/write reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out_of_range lookups are not defined. (For checked lookups see at().)

pop_back()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::pop_back ()  [inline], [constexpr], [noexcept]
```

Removes last element.
This is a typical stack operation. It shrinks the vector by one.
Note that no data is returned, and if the last element's data is needed, it should be retrieved before pop_back() is called.

push_back()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::push_back (
            const value_type & __x)  [inline], [constexpr]
```

Add data to the end of the vector.

Parameters

_↩ _x	Data to be added.

This is a typical stack operation. The function creates an element at the end of the vector and assigns the given data to it. Due to the nature of a vector this operation can be done in constant time if the vector has preallocated space available.

rbegin() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::vector< _Tp, _Alloc >::rbegin () const  [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

rbegin() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::vector< _Tp, _Alloc >::rbegin ()  [inline], [nodiscard], [constexpr], [noexcept]
```

Returns a read/write reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

rend() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::vector< _Tp, _Alloc >::rend () const  [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

rend() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::vector< _Tp, _Alloc >::rend ()  [inline], [nodiscard], [constexpr], [noexcept]
```

Returns a read/write reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

reserve()

```
template<typename _Tp, typename _Alloc>
void vector::reserve (
            size_type __n)  [constexpr]
```

Attempt to preallocate enough memory for specified number of elements.

Parameters

__n	Number of elements required.

Exceptions

std::length_error	If *n* exceeds `max_size()`.

This function attempts to reserve enough memory for the vector to hold the specified number of elements. If the number requested is more than max_size(), length_error is thrown.
The advantage of this function is that if optimal code is a necessity and the user can determine the number of elements that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of vector data.
References std::_Destroy(), _M_allocate_and_copy(), capacity(), max_size(), and std::size().

resize() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::resize (
            size_type __new_size)  [inline], [constexpr]
```

Resizes the vector to the specified number of elements.

Parameters

__new_size	Number of elements the vector should contain.

This function will resize the vector to the specified number of elements. If the number is smaller than the vector's current size the vector is truncated, otherwise default constructed elements are appended.
Referenced by __gnu_parallel::__shrink_and_double(), and __gnu_parallel::multiway_merge_exact_splitting().

resize() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::resize (
            size_type __new_size,
            const value_type & __x)  [inline], [constexpr]
```

Resizes the vector to the specified number of elements.

Parameters

__new_size	Number of elements the vector should contain.
__x	Data with which new elements should be populated.

This function will resize the vector to the specified number of elements. If the number is smaller than the vector's current size the vector is truncated, otherwise the vector is extended and new elements are populated with given data.

shrink_to_fit()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::shrink_to_fit ()  [inline], [constexpr]
```

A non-binding request to reduce capacity() to size().

size()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::vector< _Tp, _Alloc >::size () const  [inline], [nodiscard], [constexpr], [noexcept]
```

Returns the number of elements in the vector.
Referenced by __gnu_parallel::__shrink(), __gnu_parallel::__shrink_and_double(), std::vector< _Tp, polymorphic_allocator< _Tp > >::_
__gnu_parallel::list_partition(), operator=(), std::operator==(), std::vector< _Tp, polymorphic_allocator< _Tp > >::resize(),
and std::vector< _Tp, polymorphic_allocator< _Tp > >::resize().

swap()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::swap (
            vector< _Tp, _Alloc > & __x)  [inline], [constexpr], [noexcept]
```

Swaps data with another vector.

Parameters

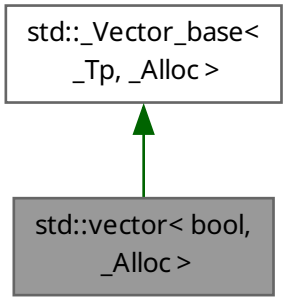
_↩ _x	A vector of the same element and allocator types.

This exchanges the elements between two vectors in constant time. (Three pointers, so it should be quite fast.) Note that the global std::swap() function is specialized such that std::swap(v1,v2) will feed to this function.
Whether the allocators are swapped depends on the allocator traits.
The documentation for this class was generated from the following files:

- stl_vector.h
- vector.tcc

5.1010 std::vector< bool, _Alloc > Class Template Reference

`#include <vector>`
Inheritance diagram for std::vector< bool, _Alloc >:



Public Types

- typedef _Alloc **allocator_type**
- typedef _Alloc **allocator_type**
- typedef _Bit_const_iterator **const_iterator**
- typedef __gnu_cxx::__normal_iterator< const_pointer, vector > **const_iterator**
- typedef const bool * **const_pointer**
- typedef _Alloc_traits::const_pointer **const_pointer**
- typedef bool **const_reference**
- typedef _Alloc_traits::const_reference **const_reference**
- typedef std::reverse_iterator< const_iterator > **const_reverse_iterator**
- typedef std::reverse_iterator< const_iterator > **const_reverse_iterator**
- typedef ptrdiff_t **difference_type**
- typedef ptrdiff_t **difference_type**
- typedef _Bit_iterator **iterator**
- typedef __gnu_cxx::__normal_iterator< pointer, vector > **iterator**
- typedef _Bit_reference * **pointer**
- typedef _Base::pointer **pointer**
- typedef _Bit_reference **reference**
- typedef _Alloc_traits::reference **reference**
- typedef std::reverse_iterator< iterator > **reverse_iterator**
- typedef std::reverse_iterator< iterator > **reverse_iterator**
- typedef size_t **size_type**
- typedef size_t **size_type**
- typedef bool **value_type**
- typedef bool **value_type**

Public Member Functions

- vector ()=default
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  constexpr **vector** (_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type())
- constexpr vector (_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type())
- constexpr **vector** (const allocator_type &__a) noexcept
- constexpr vector (const allocator_type &__a) noexcept
- constexpr **vector** (const vector &__x)
- constexpr vector (const vector &__x)
- constexpr **vector** (const vector &__x, const __type_identity_t< allocator_type > &__a)
- constexpr vector (const vector &__x, const __type_identity_t< allocator_type > &__a)
- constexpr **vector** (initializer_list< bool > __l, const allocator_type &__a=allocator_type())
- constexpr vector (initializer_list< value_type > __l, const allocator_type &__a=allocator_type())
- constexpr **vector** (size_type __n, const allocator_type &__a=allocator_type())
- constexpr vector (size_type __n, const allocator_type &__a=allocator_type())
- constexpr **vector** (size_type __n, const bool &__value, const allocator_type &__a=allocator_type())
- constexpr vector (size_type __n, const value_type &__value, const allocator_type &__a=allocator_type())
- vector (vector &&) noexcept=default
- **vector** (vector &&)=default
- constexpr vector (vector &&__rv, const __type_identity_t< allocator_type > &__m) noexcept(noexcept(vector(std::declval< vector && >(), std::declval< const allocator_type & >(), std::declval< typename _Alloc_traits::is_always_equal >())))
- constexpr **vector** (vector &&__x, const __type_identity_t< allocator_type > &__a) noexcept(_Bit_alloc_traits::↩ _S_always_equal())
- constexpr ~vector () noexcept
- constexpr auto **_M_emplace_aux** (const_iterator __position, _Args &&... __args) -> iterator
- constexpr auto **_M_emplace_aux** (const_iterator __position, _Args &&... __args) -> iterator
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  constexpr void **assign** (_InputIterator __first, _InputIterator __last)
- constexpr void assign (_InputIterator __first, _InputIterator __last)
- constexpr void **assign** (initializer_list< bool > __l)
- constexpr void assign (initializer_list< value_type > __l)
- constexpr void **assign** (size_type __n, const bool &__x)
- constexpr void assign (size_type __n, const value_type &__val)
- constexpr reference **at** (size_type __n)
- constexpr reference at (size_type __n)
- constexpr const_reference **at** (size_type __n) const
- constexpr const_reference at (size_type __n) const
- constexpr reference **back** ()
- constexpr const_reference **back** () const
- constexpr const_reference back () const noexcept
- constexpr reference back () noexcept
- constexpr const_iterator **begin** () const noexcept
- constexpr const_iterator begin () const noexcept
- constexpr iterator **begin** () noexcept
- constexpr iterator begin () noexcept
- constexpr size_type **capacity** () const noexcept
- constexpr size_type capacity () const noexcept
- constexpr const_iterator **cbegin** () const noexcept
- constexpr const_iterator cbegin () const noexcept

- constexpr const_iterator **cend** () const noexcept
- constexpr const_iterator cend () const noexcept
- constexpr void **clear** () noexcept
- constexpr void clear () noexcept
- constexpr const_reverse_iterator **crbegin** () const noexcept
- constexpr const_reverse_iterator crbegin () const noexcept
- constexpr const_reverse_iterator **crend** () const noexcept
- constexpr const_reverse_iterator crend () const noexcept
- constexpr const bool * **data** () const noexcept
- constexpr bool * data () noexcept
- template<typename... _Args>
  constexpr iterator **emplace** (const_iterator __pos, _Args &&... __args)
- constexpr iterator emplace (const_iterator __position, _Args &&... __args)
- template<typename... _Args>
  constexpr reference **emplace_back** (_Args &&... __args)
- constexpr reference **emplace_back** (_Args &&... __args)
- constexpr bool **empty** () const noexcept
- constexpr bool empty () const noexcept
- constexpr const_iterator **end** () const noexcept
- constexpr const_iterator end () const noexcept
- constexpr iterator **end** () noexcept
- constexpr iterator end () noexcept
- constexpr iterator **erase** (const_iterator __first, const_iterator __last)
- constexpr iterator erase (const_iterator __first, const_iterator __last)
- constexpr iterator **erase** (const_iterator __position)
- constexpr iterator erase (const_iterator __position)
- constexpr void **flip** () noexcept
- constexpr reference **front** ()
- constexpr const_reference **front** () const
- constexpr const_reference front () const noexcept
- constexpr reference front () noexcept
- constexpr allocator_type **get_allocator** () const
- constexpr allocator_type get_allocator () const noexcept
- constexpr allocator_type get_allocator () const noexcept
- constexpr iterator **insert** (const_iterator __p, initializer_list< bool > __l)
- iterator **insert** (const_iterator __position)
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  constexpr iterator **insert** (const_iterator __position, _InputIterator __first, _InputIterator __last)
- constexpr iterator insert (const_iterator __position, _InputIterator __first, _InputIterator __last)
- constexpr iterator **insert** (const_iterator __position, const bool &__x)
- constexpr iterator insert (const_iterator __position, const value_type &__x)
- constexpr iterator insert (const_iterator __position, initializer_list< value_type > __l)
- constexpr iterator **insert** (const_iterator __position, size_type __n, const bool &__x)
- constexpr iterator insert (const_iterator __position, size_type __n, const value_type &__x)
- constexpr iterator insert (const_iterator __position, value_type &&__x)
- constexpr size_type **max_size** () const noexcept
- constexpr size_type max_size () const noexcept
- constexpr vector & **operator=** (const vector &__x)
- constexpr vector & operator= (const vector &__x)
- constexpr vector & **operator=** (initializer_list< bool > __l)
- constexpr vector & operator= (initializer_list< value_type > __l)

- constexpr vector & operator= (vector &&__x) noexcept(_Alloc_traits::_S_nothrow_move())
- constexpr vector & **operator=** (vector &&__x) noexcept(_Bit_alloc_traits::_S_nothrow_move())
- constexpr reference **operator[ ]** (size_type __n)
- constexpr const_reference **operator[ ]** (size_type __n) const
- constexpr const_reference operator[ ] (size_type __n) const noexcept
- constexpr reference operator[ ] (size_type __n) noexcept
- constexpr void **pop_back** ()
- constexpr void pop_back () noexcept
- constexpr void **push_back** (bool __x)
- constexpr void push_back (const value_type &__x)
- constexpr void **push_back** (value_type &&__x)
- constexpr const_reverse_iterator **rbegin** () const noexcept
- constexpr const_reverse_iterator rbegin () const noexcept
- constexpr reverse_iterator **rbegin** () noexcept
- constexpr reverse_iterator rbegin () noexcept
- constexpr const_reverse_iterator **rend** () const noexcept
- constexpr const_reverse_iterator rend () const noexcept
- constexpr reverse_iterator **rend** () noexcept
- constexpr reverse_iterator rend () noexcept
- constexpr void **reserve** (size_type __n)
- constexpr void reserve (size_type __n)
- constexpr void resize (size_type __new_size)
- constexpr void **resize** (size_type __new_size, bool __x=bool())
- constexpr void resize (size_type __new_size, const value_type &__x)
- constexpr void **shrink_to_fit** ()
- constexpr void shrink_to_fit ()
- constexpr size_type **size** () const noexcept
- constexpr size_type size () const noexcept
- constexpr void **swap** (vector &__x) noexcept
- constexpr void swap (vector &__x) noexcept

Static Public Member Functions

- static constexpr void **swap** (reference __x, reference __y) noexcept

Protected Types

- typedef __gnu_cxx::__alloc_traits< _Alloc >::template rebind< _Bit_type >::other **_Bit_alloc_type**

Protected Member Functions

- constexpr pointer **_M_allocate** (size_t __n)
- constexpr pointer **_M_allocate** (size_t __n)
- constexpr pointer _M_allocate_and_copy (size_type __n, _ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator>
  constexpr void **_M_assign_aux** (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- constexpr void **_M_assign_aux** (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- template<typename _InputIterator>
  constexpr void **_M_assign_aux** (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- constexpr void **_M_assign_aux** (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- constexpr void **_M_assign_dispatch** (_InputIterator __first, _InputIterator __last, __false_type)
- constexpr void **_M_assign_dispatch** (_Integer __n, _Integer __val, __true_type)

- constexpr size_type **_M_check_len** (size_type __n, const char *__s) const
- constexpr size_type **_M_check_len** (size_type __n, const char *__s) const
- constexpr iterator **_M_copy_aligned** (const_iterator __first, const_iterator __last, iterator __result)
- constexpr void **_M_create_storage** (size_t __n)
- constexpr void **_M_deallocate** ()
- constexpr void **_M_deallocate** (pointer __p, size_t __n)
- constexpr void **_M_deallocate** (pointer __p, size_t __n)
- constexpr void **_M_deallocate** (pointer __p, size_t __n)
- constexpr void **_M_default_append** (size_type __n)
- constexpr void **_M_default_initialize** (size_type __n)
- constexpr iterator **_M_emplace_aux** (const_iterator __position, _Args &&... __args)
- constexpr iterator **_M_emplace_aux** (const_iterator __position, value_type &&__v)
- constexpr iterator **_M_erase** (iterator __first, iterator __last)
- constexpr iterator **_M_erase** (iterator __first, iterator __last)
- constexpr iterator **_M_erase** (iterator __pos)
- constexpr iterator **_M_erase** (iterator __position)
- constexpr void **_M_erase_at_end** (iterator __pos)
- constexpr void **_M_erase_at_end** (pointer __pos) noexcept
- constexpr void **_M_fill_append** (size_type __n, const value_type &__x)
- constexpr void **_M_fill_assign** (size_t __n, bool __x)
- constexpr void **_M_fill_assign** (size_type __n, const value_type &__val)
- constexpr void **_M_fill_initialize** (size_type __n, const value_type &__value)
- constexpr void **_M_fill_insert** (iterator __pos, size_type __n, const value_type &__x)
- constexpr void **_M_fill_insert** (iterator __position, size_type __n, bool __x)
- constexpr const _Bit_alloc_type & **_M_get_Bit_allocator** () const noexcept
- constexpr _Bit_alloc_type & **_M_get_Bit_allocator** () noexcept
- constexpr const _Tp_alloc_type & **_M_get_Tp_allocator** () const noexcept
- constexpr const _Tp_alloc_type & **_M_get_Tp_allocator** () const noexcept
- constexpr _Tp_alloc_type & **_M_get_Tp_allocator** () noexcept
- constexpr _Tp_alloc_type & **_M_get_Tp_allocator** () noexcept
- constexpr void **_M_initialize** (size_type __n)
- template<typename _ForwardIterator>
  constexpr void **_M_initialize_range** (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- template<typename _InputIterator>
  constexpr void **_M_initialize_range** (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- constexpr void **_M_initialize_value** (bool __x) noexcept
- constexpr void **_M_insert_aux** (iterator __position, _Arg &&__arg)
- constexpr void **_M_insert_aux** (iterator __position, bool __x)
- constexpr void **_M_insert_dispatch** (iterator __pos, _InputIterator __first, _InputIterator __last, __false_type)
- constexpr void **_M_insert_dispatch** (iterator __pos, _Integer __n, _Integer __val, __true_type)
- template<typename _InputIterator>
  constexpr void **_M_insert_range** (iterator __pos, _InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- template<typename _ForwardIterator>
  constexpr void **_M_insert_range** (iterator __position, _ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- constexpr iterator **_M_insert_rval** (const_iterator __position, value_type &&__v)
- constexpr void **_M_move_data** (_Bvector_base &&__x) noexcept
- constexpr void **_M_range_check** (size_type __n) const
- constexpr void _M_range_check (size_type __n) const
- constexpr void **_M_range_initialize** (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- constexpr void **_M_range_initialize** (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)

- constexpr void **_M_range_initialize_n** (_Iterator __first, _Sentinel __last, size_type __n)
- constexpr void **_M_range_insert** (iterator __pos, _ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- constexpr void **_M_range_insert** (iterator __pos, _InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- constexpr void **_M_realloc_append** (_Args &&... __args)
- constexpr void **_M_realloc_insert** (iterator __position, _Args &&... __args)
- constexpr void **_M_reallocate** (size_type __n)
- constexpr bool **_M_shrink_to_fit** ()
- constexpr bool **_M_shrink_to_fit** ()
- void **data** ()=delete

Static Protected Member Functions

- static constexpr size_type **_S_check_init_len** (size_type __n, const allocator_type &__a)
- static constexpr size_type **_S_max_size** (const _Tp_alloc_type &__a) noexcept
- static constexpr size_t **_S_nword** (size_t __n)

Protected Attributes

- _Vector_impl **_M_impl**
- _Vector_impl **_M_impl**

Friends

- struct **std::hash**< **vector** >

5.1010.1 Detailed Description

template<**typename _Alloc**>
class std::vector< **bool, _Alloc** >

A specialization of vector for booleans which offers fixed time access to individual elements in any order.

Since

C++98

Template Parameters

_Alloc	Allocator type.

Note that vector<bool> does not actually meet the requirements for being a container. This is because the reference and pointer types are not really references and pointers to bool. See DR96 for details.

See also

vector for function documentation.

In some terminology a vector can be described as a dynamic C-style array, it offers fast and efficient access to individual elements in any order and saves the user from worrying about memory and size allocation. Subscripting ([]) access is also provided as with C-style arrays.

5.1010.2 Constructor & Destructor Documentation

vector() [1/10]

```
std::vector< bool, _Alloc >::vector ()  [default]
```

Creates a vector with no elements.

vector() [2/10]

```
std::vector< bool, _Alloc >::vector (
            const allocator_type & __a)  [inline], [explicit], [constexpr], [noexcept]
```

Creates a vector with no elements.

Parameters

_↩ _a	An allocator object.

vector() [3/10]

```
std::vector< bool, _Alloc >::vector (
            size_type __n,
            const allocator_type & __a = allocator_type())  [inline], [explicit], [constexpr]
```

Creates a vector with default constructed elements.

Parameters

_↩ _n	The number of elements to initially create.
_↩ _a	An allocator.

This constructor fills the vector with __*n* default constructed elements.

vector() [4/10]

```
std::vector< bool, _Alloc >::vector (
            size_type __n,
            const value_type & __value,
            const allocator_type & __a = allocator_type())  [inline], [constexpr]
```

Creates a vector with copies of an exemplar element.

Parameters

__*n*	The number of elements to initially create.
__*value*	An element to copy.
__*a*	An allocator.

This constructor fills the vector with __*n* copies of __*value*.

vector() [5/10]

```
std::vector< bool, _Alloc >::vector (
            const vector< bool, _Alloc > & __x)  [inline], [constexpr]
```

Vector copy constructor.

Parameters

_↩ _x	A vector of identical element and allocator types.

All the elements of __x are copied, but any unused capacity in __x will not be copied (i.e. capacity() == size() in the new vector).
The newly-created vector uses a copy of the allocator object used by __x (unless the allocator traits dictate a different object).

vector() [6/10]

```
std::vector< bool, _Alloc >::vector (
            vector< bool, _Alloc > && )  [default], [noexcept]
```

Vector move constructor.
The newly-created vector contains the exact contents of the moved instance. The contents of the moved instance are a valid, but unspecified vector.

vector() [7/10]

```
std::vector< bool, _Alloc >::vector (
            const vector< bool, _Alloc > & __x,
            const __type_identity_t< allocator_type > & __a)  [inline], [constexpr]
```

Copy constructor with alternative allocator.

vector() [8/10]

```
std::vector< bool, _Alloc >::vector (
            vector< bool, _Alloc > && __rv,
            const __type_identity_t< allocator_type > & __m)  [inline], [constexpr], [noexcept]
```

Move constructor with alternative allocator.

vector() [9/10]

```
std::vector< bool, _Alloc >::vector (
            initializer_list< value_type > __l,
            const allocator_type & __a = allocator_type())  [inline], [constexpr]
```

Builds a vector from an initializer list.

Parameters

__l	An initializer_list.
__a	An allocator.

Create a vector consisting of copies of the elements in the initializer_list __l.
This will call the element type's copy constructor N times (where N is __l.size()) and do no memory reallocation.

vector() [10/10]

```
std::vector< bool, _Alloc >::vector (
            _InputIterator __first,
            _InputIterator __last,
            const allocator_type & __a = allocator_type())  [inline], [constexpr]
```

Builds a vector from a range.

Parameters

__first	An input iterator.
__last	An input iterator.
__a	An allocator.

Create a vector consisting of copies of the elements from [first,last).
If the iterators are forward, bidirectional, or random-access, then this will call the elements' copy constructor N times (where N is distance(first,last)) and do no memory reallocation. But if only input iterators are used, then this will do at most 2N calls to the copy constructor, and logN memory reallocations.

~vector()

```
std::vector< bool, _Alloc >::~vector ()  [inline], [constexpr], [noexcept]
```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

5.1010.3 Member Function Documentation

_M_allocate_and_copy()

```
pointer std::vector< bool, _Alloc >::_M_allocate_and_copy (
            size_type __n,
            _ForwardIterator __first,
            _ForwardIterator __last)  [inline], [constexpr], [protected]
```

Memory expansion handler. Uses the member allocation function to obtain *n* bytes of memory, and then copies [first,last) into it.

_M_range_check()

```
void std::vector< bool, _Alloc >::_M_range_check (
            size_type __n) const  [inline], [constexpr], [protected]
```

Safety check used only from at().

assign() [1/3]

```
void std::vector< bool, _Alloc >::assign (
            _InputIterator __first,
            _InputIterator __last)  [inline], [constexpr]
```

Assigns a range to a vector.

Parameters

__first	An input iterator.
__last	An input iterator.

This function fills a vector with copies of the elements in the range [__first,__last).
Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned.

assign() [2/3]

```
void std::vector< bool, _Alloc >::assign (
            initializer_list< value_type > __l)  [inline], [constexpr]
```

Assigns an initializer list to a vector.

Parameters

↩ _↩ ↩ _↩ *l*	An initializer_list.

This function fills a vector with copies of the elements in the initializer list *__l*.
Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned.

assign() [3/3]

```
void std::vector< bool, _Alloc >::assign (
            size_type __n,
            const value_type & __val)  [inline], [constexpr]
```

Assigns a given value to a vector.

Parameters

__n	Number of elements to be assigned.
__val	Value to be assigned.

This function fills a vector with *__n* copies of the given value. Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned.

at() [1/2]

```
reference std::vector< bool, _Alloc >::at (
            size_type __n)  [inline], [nodiscard], [constexpr]
```

Provides access to the data contained in the vector.

Parameters

_↩ *_n*	The index of the element for which data should be accessed.

Returns

Read/write reference to data.

Exceptions

std::out_of_range	If *__n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the vector. The function throws out_of_range if the check fails.

at() [2/2]

```
const_reference std::vector< bool, _Alloc >::at (
            size_type __n) const  [inline], [nodiscard], [constexpr]
```

Provides access to the data contained in the vector.

Parameters

_↩ _n	The index of the element for which data should be accessed.

Returns

Read-only (constant) reference to data.

Exceptions

std::out_of_range	If *__n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the vector. The function throws out_of_range if the check fails.

back() [1/2]

`const_reference std::vector< bool, _Alloc >::back () const [inline], [nodiscard], [constexpr], [noexcept]`
Returns a read-only (constant) reference to the data at the last element of the vector.

back() [2/2]

`reference std::vector< bool, _Alloc >::back () [inline], [nodiscard], [constexpr], [noexcept]`
Returns a read/write reference to the data at the last element of the vector.

begin() [1/2]

`const_iterator std::vector< bool, _Alloc >::begin () const [inline], [nodiscard], [constexpr], [noexcept]`
Returns a read-only (constant) iterator that points to the first element in the vector. Iteration is done in ordinary element order.

begin() [2/2]

`iterator std::vector< bool, _Alloc >::begin () [inline], [nodiscard], [constexpr], [noexcept]`
Returns a read/write iterator that points to the first element in the vector. Iteration is done in ordinary element order.

capacity()

`size_type std::vector< bool, _Alloc >::capacity () const [inline], [nodiscard], [constexpr], [noexcept]`
Returns the total number of elements that the vector can hold before needing to allocate more memory.

cbegin()

`const_iterator std::vector< bool, _Alloc >::cbegin () const [inline], [constexpr], [noexcept]`
Returns a read-only (constant) iterator that points to the first element in the vector. Iteration is done in ordinary element order.

cend()

`const_iterator std::vector< bool, _Alloc >::cend () const [inline], [constexpr], [noexcept]`
Returns a read-only (constant) iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

clear()

```
void std::vector< bool, _Alloc >::clear ()  [inline], [constexpr], [noexcept]
```
Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

crbegin()

```
const_reverse_iterator std::vector< bool, _Alloc >::crbegin () const  [inline], [constexpr],
[noexcept]
```
Returns a read-only (constant) reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

crend()

```
const_reverse_iterator std::vector< bool, _Alloc >::crend () const  [inline], [constexpr], [noexcept]
```
Returns a read-only (constant) reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

data()

```
bool * std::vector< bool, _Alloc >::data ()  [inline], [nodiscard], [constexpr], [noexcept]
```
Returns a pointer such that [data(), data() + size()) is a valid range. For a non-empty vector, data() == &front().

emplace()

```
iterator std::vector< bool, _Alloc >::emplace (
            const_iterator __position,
            _Args &&... __args)  [inline], [constexpr]
```
Inserts an object in vector before specified iterator.

Parameters

__position	A const_iterator into the vector.
__args	Arguments.

Returns

An iterator that points to the inserted data.

This function will insert an object of type T constructed with T(std::forward<Args>(args)...) before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using std::list.

empty()

```
bool std::vector< bool, _Alloc >::empty () const  [inline], [nodiscard], [constexpr], [noexcept]
```
Returns true if the vector is empty. (Thus begin() would equal end().)

end() [1/2]

```
const_iterator std::vector< bool, _Alloc >::end () const  [inline], [nodiscard], [constexpr],
[noexcept]
```
Returns a read-only (constant) iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

end() [2/2]

```
iterator std::vector< bool, _Alloc >::end () [inline], [nodiscard], [constexpr], [noexcept]
```

Returns a read/write iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

erase() [1/2]

```
iterator std::vector< bool, _Alloc >::erase (
            const_iterator __first,
            const_iterator __last) [inline], [constexpr]
```

Remove a range of elements.

Parameters

__first	Iterator pointing to the first element to be erased.
__last	Iterator pointing to one past the last element to be erased.

Returns

An iterator pointing to the element pointed to by *__last* prior to erasing (or end()).

This function will erase the elements in the range [__first,__last) and shorten the vector accordingly.
Note This operation could be expensive and if it is frequently used the user should consider using std::list. The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [2/2]

```
iterator std::vector< bool, _Alloc >::erase (
            const_iterator __position) [inline], [constexpr]
```

Remove element at given position.

Parameters

__position	Iterator pointing to element to be erased.

Returns

An iterator pointing to the next element (or end()).

This function will erase the element at the given position and thus shorten the vector by one.
Note This operation could be expensive and if it is frequently used the user should consider using std::list. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

front() [1/2]

```
const_reference std::vector< bool, _Alloc >::front () const [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read-only (constant) reference to the data at the first element of the vector.

front() [2/2]

```
reference std::vector< bool, _Alloc >::front () [inline], [nodiscard], [constexpr], [noexcept]
```

Returns a read/write reference to the data at the first element of the vector.

get_allocator() [1/2]

```
() const
```

Get a copy of the memory allocation object.

get_allocator() [2/2]

```
allocator_type std::_Vector_base< bool, _Alloc >::get_allocator () const  [inline], [constexpr],
[noexcept]
```

Get a copy of the memory allocation object.

insert() [1/5]

```
iterator std::vector< bool, _Alloc >::insert (
            const_iterator __position,
            _InputIterator __first,
            _InputIterator __last)  [inline], [constexpr]
```

Inserts a range into the vector.

Parameters

__position	A const_iterator into the vector.
__first	An input iterator.
__last	An input iterator.

Returns

An iterator that points to the inserted data.

This function will insert copies of the data in the range [__first,__last) into the vector before the location specified by *pos*. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using std::list.

insert() [2/5]

```
vector< bool, _Alloc >::iterator vector::insert (
            const_iterator __position,
            const value_type & __x)  [constexpr]
```

Inserts given value into vector before specified iterator.

Parameters

__position	A const_iterator into the vector.
__x	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using std::list.

insert() [3/5]

```
iterator std::vector< bool, _Alloc >::insert (
            const_iterator __position,
            initializer_list< value_type > __l)  [inline], [constexpr]
```

Inserts an initializer_list into the vector.

Parameters

__position	An iterator into the vector.
__l	An initializer_list.

This function will insert copies of the data in the initializer_list *l* into the vector before the location specified by *position*. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using std::list.

insert() [4/5]

```
iterator std::vector< bool, _Alloc >::insert (
            const_iterator __position,
            size_type __n,
            const value_type & __x)  [inline], [constexpr]
```

Inserts a number of copies of given data into the vector.

Parameters

__position	A const_iterator into the vector.
__n	Number of elements to be inserted.
__x	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a specified number of copies of the given data before the location specified by *position*. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using std::list.

insert() [5/5]

```
iterator std::vector< bool, _Alloc >::insert (
            const_iterator __position,
            value_type && __x)  [inline], [constexpr]
```

Inserts given rvalue into vector before specified iterator.

Parameters

__position	A const_iterator into the vector.
__x	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using std::list.

max_size()

```
size_type std::vector< bool, _Alloc >::max_size () const  [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns the size() of the largest possible vector.

operator=() [1/3]

```
vector< bool, _Alloc > & vector::operator= (
            const vector< bool, _Alloc > & __x)  [constexpr]
```

Vector assignment operator.

Parameters

_↩ _*x*	A vector of identical element and allocator types.

All the elements of __*x* are copied, but any unused capacity in __*x* will not be copied.
Whether the allocator is copied depends on the allocator traits.

operator=() [2/3]

```
vector & std::vector< bool, _Alloc >::operator= (
            initializer_list< value_type > __l)  [inline], [constexpr]
```

Vector list assignment operator.

Parameters

↩ _↩ ↩ _↩ *l*	An initializer_list.

This function fills a vector with copies of the elements in the initializer list __*l*.
Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned.

operator=() [3/3]

```
vector & std::vector< bool, _Alloc >::operator= (
            vector< bool, _Alloc > && __x)  [inline], [constexpr], [noexcept]
```

Vector move assignment operator.

Parameters

_↩ _*x*	A vector of identical element and allocator types.

The contents of __*x* are moved into this vector (without copying, if the allocators permit it). Afterwards __*x* is a valid, but unspecified vector.
Whether the allocator is moved depends on the allocator traits.

operator[]() [1/2]

```
const_reference std::vector< bool, _Alloc >::operator[] (
            size_type __n) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Subscript access to the data contained in the vector.

Parameters

_↩ _n	The index of the element for which data should be accessed.

Returns

Read-only (constant) reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out_of_range lookups are not defined. (For checked lookups see at().)

operator[]() [2/2]

```
reference std::vector< bool, _Alloc >::operator[] (
            size_type __n)  [inline], [nodiscard], [constexpr], [noexcept]
```

Subscript access to the data contained in the vector.

Parameters

_↩ _n	The index of the element for which data should be accessed.

Returns

Read/write reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out_of_range lookups are not defined. (For checked lookups see at().)

pop_back()

```
void std::vector< bool, _Alloc >::pop_back ()  [inline], [constexpr], [noexcept]
```

Removes last element.
This is a typical stack operation. It shrinks the vector by one.
Note that no data is returned, and if the last element's data is needed, it should be retrieved before pop_back() is called.

push_back()

```
void std::vector< bool, _Alloc >::push_back (
            const value_type & __x)  [inline], [constexpr]
```

Add data to the end of the vector.

Parameters

_↩ _x	Data to be added.

This is a typical stack operation. The function creates an element at the end of the vector and assigns the given data to it. Due to the nature of a vector this operation can be done in constant time if the vector has preallocated space available.

rbegin() [1/2]

```
const_reverse_iterator std::vector< bool, _Alloc >::rbegin () const  [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

rbegin() [2/2]

```
reverse_iterator std::vector< bool, _Alloc >::rbegin ()  [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read/write reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

rend() [1/2]

```
const_reverse_iterator std::vector< bool, _Alloc >::rend () const  [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

rend() [2/2]

```
reverse_iterator std::vector< bool, _Alloc >::rend ()  [inline], [nodiscard], [constexpr], [noexcept]
```

Returns a read/write reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

reserve()

```
void vector::reserve (
            size_type __n)  [constexpr]
```

Attempt to preallocate enough memory for specified number of elements.

Parameters

_↩ _*n*	Number of elements required.

Exceptions

std::length_error	If *n* exceeds `max_size()`.

This function attempts to reserve enough memory for the vector to hold the specified number of elements. If the number requested is more than max_size(), length_error is thrown.
The advantage of this function is that if optimal code is a necessity and the user can determine the number of elements that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of vector data.

resize() [1/2]

```
void std::vector< bool, _Alloc >::resize (
            size_type __new_size)  [inline], [constexpr]
```

Resizes the vector to the specified number of elements.

Parameters

__new_size	Number of elements the vector should contain.

This function will resize the vector to the specified number of elements. If the number is smaller than the vector's current size the vector is truncated, otherwise default constructed elements are appended.

resize() [2/2]

```
void std::vector< bool, _Alloc >::resize (
            size_type __new_size,
            const value_type & __x)  [inline], [constexpr]
```

Resizes the vector to the specified number of elements.

Parameters

__new_size	Number of elements the vector should contain.
__x	Data with which new elements should be populated.

This function will resize the vector to the specified number of elements. If the number is smaller than the vector's current size the vector is truncated, otherwise the vector is extended and new elements are populated with given data.

shrink_to_fit()

```
void std::vector< bool, _Alloc >::shrink_to_fit ()  [inline], [constexpr]
```

A non-binding request to reduce capacity() to size().

size()

```
size_type std::vector< bool, _Alloc >::size () const  [inline], [nodiscard], [constexpr], [noexcept]
```

Returns the number of elements in the vector.

swap()

```
void std::vector< bool, _Alloc >::swap (
            vector< bool, _Alloc > & __x)  [inline], [constexpr], [noexcept]
```

Swaps data with another vector.

Parameters

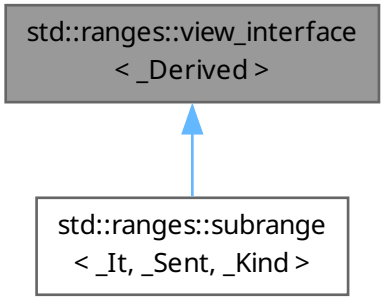
__x	A vector of the same element and allocator types.

This exchanges the elements between two vectors in constant time. (Three pointers, so it should be quite fast.) Note that the global std::swap() function is specialized such that std::swap(v1,v2) will feed to this function.
Whether the allocators are swapped depends on the allocator traits.
The documentation for this class was generated from the following files:

- stl_bvector.h
- vector.tcc

5.1011 std::ranges::view_interface< _Derived > Class Template Reference

`#include <ranges_util.h>`
Inheritance diagram for std::ranges::view_interface< _Derived >:



Public Member Functions

- constexpr decltype(auto) **back** ()
- constexpr decltype(auto) **back** () const
- constexpr auto **data** () const noexcept(noexcept(ranges::begin(_M_derived())))
- constexpr auto **data** () noexcept(noexcept(ranges::begin(_M_derived())))
- constexpr bool **empty** () const noexcept(noexcept(_S_empty(_M_derived())))
- constexpr bool **empty** () const noexcept(noexcept(ranges::size(_M_derived())==0))
- constexpr bool **empty** () noexcept(noexcept(_S_empty(_M_derived())))
- constexpr bool **empty** () noexcept(noexcept(ranges::size(_M_derived())==0))
- constexpr decltype(auto) **front** ()
- constexpr decltype(auto) **front** () const
- constexpr **operator bool** () const noexcept(noexcept(ranges::empty(_M_derived())))
- constexpr **operator bool** () noexcept(noexcept(ranges::empty(_M_derived())))
- template<random_access_range _Range = _Derived>
  constexpr decltype(auto) **operator[ ]** (range_difference_t< _Range > __n)
- template<random_access_range _Range = const _Derived>
  constexpr decltype(auto) **operator[ ]** (range_difference_t< _Range > __n) const
- constexpr auto **size** () const noexcept(noexcept(_S_size(_M_derived())))
- constexpr auto **size** () noexcept(noexcept(_S_size(_M_derived())))

5.1011.1 Detailed Description

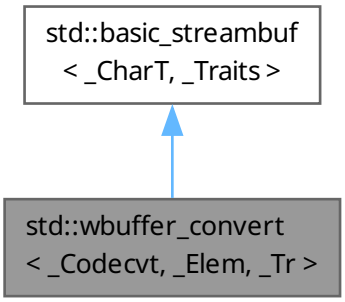
template<typename _Derived>
requires is_class_v<_Derived> && same_as<_Derived, remove_cv_t<_Derived>>
class std::ranges::view_interface< _Derived >

The ranges::view_interface class template.
The documentation for this class was generated from the following file:

- ranges_util.h

5.1012 std::wbuffer_convert< _Codecvt, _Elem, _Tr > Class Template Reference

`#include <locale_conv.h>`
Inheritance diagram for std::wbuffer_convert< _Codecvt, _Elem, _Tr >:



Public Types

- typedef _Codecvt::state_type **state_type**

- typedef _CharT char_type
- typedef _Traits traits_type
- typedef traits_type::int_type int_type
- typedef traits_type::pos_type pos_type
- typedef traits_type::off_type off_type

- typedef basic_streambuf< char_type, traits_type > __streambuf_type

Public Member Functions

- wbuffer_convert ()
- **wbuffer_convert** (const wbuffer_convert &)=delete
- wbuffer_convert (streambuf *__bytebuf, _Codecvt *__pcvt=new _Codecvt, state_type __state=state_type())
- locale getloc () const
- streamsize in_avail ()
- wbuffer_convert & **operator=** (const wbuffer_convert &)=delete
- locale pubimbue (const locale &__loc)
- streambuf * **rdbuf** () const noexcept
- streambuf * **rdbuf** (streambuf *__bytebuf) noexcept
- int_type sbumpc ()
- int_type sgetc ()
- streamsize sgetn (char_type *__s, streamsize __n)
- int_type snextc ()
- int_type sputbackc (char_type __c)
- int_type sputc (char_type __c)

- streamsize sputn (const char_type *__s, streamsize __n)
- state_type state () const noexcept
- int_type sungetc ()

- basic_streambuf * pubsetbuf (char_type *__s, streamsize __n)
- pos_type pubseekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)
- pos_type pubseekpos (pos_type __sp, ios_base::openmode __mode=ios_base::in|ios_base::out)
- int pubsync ()

Protected Member Functions

- void **__safe_gbump** (streamsize __n)
- void **__safe_pbump** (streamsize __n)
- void gbump (int __n)
- virtual void imbue (const locale &__loc)
- _Wide_streambuf::int_type overflow (typename _Wide_streambuf::int_type __out)
- virtual int_type pbackfail (int_type __c=traits_type::eof())
- void pbump (int __n)
- virtual pos_type seekoff (off_type, ios_base::seekdir, ios_base::openmode=ios_base::in|ios_base::out)
- virtual pos_type seekpos (pos_type, ios_base::openmode=ios_base::in|ios_base::out)
- virtual basic_streambuf< char_type, _Traits > * setbuf (char_type *, streamsize)
- void setg (char_type *__gbeg, char_type *__gnext, char_type *__gend)
- void setp (char_type *__pbeg, char_type *__pend)
- virtual streamsize showmanyc ()
- void **swap** (basic_streambuf &__sb)
- int sync ()
- virtual int_type uflow ()
- _Wide_streambuf::int_type underflow ()
- virtual streamsize xsgetn (char_type *__s, streamsize __n)
- virtual streamsize xsputn (const char_type *__s, streamsize __n)
- streamsize **xsputn** (const typename _Wide_streambuf::char_type *__s, streamsize __n)

- char_type * eback () const
- char_type * gptr () const
- char_type * egptr () const

- char_type * pbase () const
- char_type * pptr () const
- char_type * epptr () const

Protected Attributes

- locale _M_buf_locale
- char_type * _M_in_beg
- char_type * _M_in_cur
- char_type * _M_in_end
- char_type * _M_out_beg
- char_type * _M_out_cur
- char_type * _M_out_end

5.1012.1 Detailed Description

template<typename _Codecvt, typename _Elem = wchar_t, typename _Tr = char_traits<_Elem>>
class std::wbuffer_convert< _Codecvt, _Elem, _Tr >

Buffer conversions.

5.1012.2 Member Typedef Documentation

__streambuf_type

```
template<typename _CharT, typename _Traits>
typedef basic_streambuf<char_type, traits_type> std::basic_streambuf< _CharT, _Traits >::__↩
streambuf_type  [inherited]
```

This is a non-standard type.

char_type

```
template<typename _CharT, typename _Traits>
typedef _CharT std::basic_streambuf< _CharT, _Traits >::char_type  [inherited]
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

int_type

```
template<typename _CharT, typename _Traits>
typedef traits_type::int_type std::basic_streambuf< _CharT, _Traits >::int_type  [inherited]
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

off_type

```
template<typename _CharT, typename _Traits>
typedef traits_type::off_type std::basic_streambuf< _CharT, _Traits >::off_type  [inherited]
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

pos_type

```
template<typename _CharT, typename _Traits>
typedef traits_type::pos_type std::basic_streambuf< _CharT, _Traits >::pos_type  [inherited]
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

traits_type

```
template<typename _CharT, typename _Traits>
typedef _Traits std::basic_streambuf< _CharT, _Traits >::traits_type  [inherited]
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

5.1012.3 Constructor & Destructor Documentation

wbuffer_convert() [1/2]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Tr = char_traits<_Elem>>
std::wbuffer_convert< _Codecvt, _Elem, _Tr >::wbuffer_convert ()  [inline]
```

Default constructor.
References wbuffer_convert().
Referenced by wbuffer_convert().

wbuffer_convert() [2/2]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Tr = char_traits<_Elem>>
std::wbuffer_convert< _Codecvt, _Elem, _Tr >::wbuffer_convert (
            streambuf * __bytebuf,
            _Codecvt * __pcvt = new _Codecvt,
            state_type __state = state_type())  [inline], [explicit]
```

Constructor.

Parameters

__bytebuf	The underlying byte stream buffer.
__pcvt	The facet to use for conversions.
__state	Initial conversion state.

Takes ownership of __pcvt and will delete it in the destructor.
References std::basic_streambuf< _CharT, _Traits >::setg(), and std::basic_streambuf< _CharT, _Traits >::setp().

5.1012.4 Member Function Documentation

eback()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::eback () const  [inline], [protected], [inherited]
```

Access to the get area.
These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_stringbuf< _Cha std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_filebuf< _CharT, _Traits >::underflow(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

egptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::egptr () const  [inline], [protected], [inherited]
```

Access to the get area.
These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Referenced by std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_filebuf< _CharT, _Traits >::underflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), std::wbuffer_convert< _Codecvt, _Elem, _Tr >::underflow(), std::basic_filebuf< _CharT, _Traits >::xsgetn(), and xsgetn().

epptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::epptr () const  [inline], [protected], [inherited]
```

Access to the put area.
These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_filebuf< _CharT, _Traits >::xsputn(), and xsputn().

gbump()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::gbump (
            int __n)  [inline], [protected], [inherited]
```

Moving the read position.

Parameters

_↩ _n	The delta by which to move.

This just advances the read position without returning any data.
Referenced by std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

getloc()

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::getloc () const  [inline], [inherited]
```

Locale access.

Returns

The current locale in effect.

If pubimbue(loc) has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

gptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::gptr () const  [inline], [protected], [inherited]
```

Access to the get area.
These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_stringbuf< _Cha std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_filebuf< _CharT, _Traits >::underflow(), std::basic_stringbuf< _CharT, _T std::wbuffer_convert< _Codecvt, _Elem, _Tr >::underflow(), std::basic_filebuf< _CharT, _Traits >::xsgetn(), and xsgetn().

imbue()

```
template<typename _CharT, typename _Traits>
virtual void std::basic_streambuf< _CharT, _Traits >::imbue (
            const locale & __loc)  [inline], [protected], [virtual], [inherited]
```

Changes translations.

Parameters

__loc	A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

Note

Base class version does nothing.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< _CharT, std::char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< wchar_t >, and std::basic_filebuf< wchar_t >.

in_avail()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::in_avail ()  [inline], [inherited]
```

Looking ahead into the stream.

Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

overflow()

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Tr = char_traits<_Elem>>
_Wide_streambuf::int_type std::wbuffer_convert< _Codecvt, _Elem, _Tr >::overflow (
            typename _Wide_streambuf::int_type __c)  [inline], [protected], [virtual]
```

Consumes data from the buffer; writes to the controlled sequence.

Parameters

__c	An additional character to consume.

Returns

eof() to indicate failure, something else (usually __*c*, or not_eof())

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if __*c* is not `eof()`.
For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.
A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References std::basic_streambuf< _CharT, _Traits >::sputc().

pbackfail()

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::pbackfail (
            int_type __c = traits_type::eof())  [inline], [protected], [virtual], [inherited]
```

Tries to back up the input sequence.

Parameters

_↩ _*c*	The character to be inserted back into the sequence.

Returns

eof() on failure, *some other value* on success

Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

Note

Base class version does nothing, returns eof().

Reimplemented in __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, std::basic_filebuf< char >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< char_type, traits_ std::basic_filebuf< wchar_t >, std::basic_filebuf< wchar_t >, std::basic_stringbuf< _CharT, _Traits, _Alloc >, std::basic_stringbuf< char >, std::basic_stringbuf< char >, std::basic_stringbuf< wchar_t >, and std::basic_stringbuf< wchar_t >.

pbase()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::pbase () const  [inline], [protected], [inherited]
```

Access to the put area.
These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, std::basic_filebuf< _CharT, _Traits >::sync(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

pbump()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::pbump (
            int __n)  [inline], [protected], [inherited]
```

Moving the write position.

Parameters

_↩ _n	The delta by which to move.

This just advances the write position without returning any data.
Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow().

pptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::pptr () const  [inline], [protected], [inherited]
```

Access to the put area.
These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_filebuf< _CharT, _T std::basic_filebuf< _CharT, _Traits >::xsputn(), and xsputn().

pubimbue()

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::pubimbue (
            const locale & __loc)  [inline], [inherited]
```

Entry point for imbue().

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls the derived imbue(__loc).

pubseekoff()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekoff (
            off_type __off,
            ios_base::seekdir __way,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [inline], [inherited]
```

Alters the stream position.

Parameters

__off	Offset.
__way	Value for ios_base::seekdir.
__mode	Value for ios_base::openmode.

Calls virtual seekoff function.

pubseekpos()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekpos (
            pos_type __sp,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [inline], [inherited]
```

Alters the stream position.

Parameters

__sp	Position
__mode	Value for ios_base::openmode.

Calls virtual seekpos function.

pubsetbuf()

```
template<typename _CharT, typename _Traits>
basic_streambuf * std::basic_streambuf< _CharT, _Traits >::pubsetbuf (
            char_type * __s,
            streamsize __n)  [inline], [inherited]
```

Entry points for derived buffer functions.
The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

pubsync()

```
template<typename _CharT, typename _Traits>
int std::basic_streambuf< _CharT, _Traits >::pubsync ()  [inline], [inherited]
```

Calls virtual sync function.
Referenced by std::basic_istream< _CharT, _Traits >::sync().

sbumpc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sbumpc ()  [inline], [inherited]
```

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.
Referenced by std::basic_istream< _CharT, _Traits >::getline().

seekoff()

```
template<typename _CharT, typename _Traits>
virtual pos_type std::basic_streambuf< _CharT, _Traits >::seekoff (
            off_type ,
            ios_base::seekdir ,
            ios_base::openmode  = ios_base::in | ios_base::out)  [inline], [protected], [virtual],
[inherited]
```

Alters the stream positions.
Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< char_type, traits_ std::basic_filebuf< wchar_t >, std::basic_filebuf< wchar_t >, std::basic_stringbuf< _CharT, _Traits, _Alloc >, std::basic_stringbuf< char >, std::basic_stringbuf< char >, std::basic_stringbuf< wchar_t >, and std::basic_stringbuf< wchar_t >.

seekpos()

```
template<typename _CharT, typename _Traits>
virtual pos_type std::basic_streambuf< _CharT, _Traits >::seekpos (
            pos_type ,
            ios_base::openmode  = ios_base::in | ios_base::out)  [inline], [protected], [virtual],
[inherited]
```

Alters the stream positions.
Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< char_type, traits_ std::basic_filebuf< wchar_t >, std::basic_filebuf< wchar_t >, std::basic_stringbuf< _CharT, _Traits, _Alloc >, std::basic_stringbuf< char >, std::basic_stringbuf< char >, std::basic_stringbuf< wchar_t >, and std::basic_stringbuf< wchar_t >.

setbuf()

```
template<typename _CharT, typename _Traits>
virtual basic_streambuf< char_type, _Traits > * std::basic_streambuf< _CharT, _Traits >::setbuf
(
            char_type * ,
            streamsize )  [inline], [protected], [virtual], [inherited]
```

Manipulates the buffer.
Each derived class provides its own appropriate behavior. See the next-to-last paragraph of https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html#io.streambuf.buffering for more on this function.

Note

Base class version does nothing, returns `this`.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< _CharT, std::char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< wchar_t >, std::basic_stringbuf< _CharT, _Traits, _Alloc >, std::basic_stringbuf< char >, and std::basic_stringbuf< char >.

setg()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setg (
             char_type * __gbeg,
             char_type * __gnext,
             char_type * __gend)  [inline], [protected], [inherited]
```

Setting the three read area pointers.

Parameters

__gbeg	A pointer.
__gnext	A pointer.
__gend	A pointer.

Postcondition

__gbeg == `eback()`, *__gnext* == `gptr()`, and *__gend* == `egptr()`

Referenced by std::wbuffer_convert< _Codecvt, _Elem, _Tr >::wbuffer_convert(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflo std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

setp()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setp (
             char_type * __pbeg,
             char_type * __pend)  [inline], [protected], [inherited]
```

Setting the three write area pointers.

Parameters

__pbeg	A pointer.
__pend	A pointer.

Postcondition

__pbeg == `pbase()`, *__pbeg* == `pptr()`, and *__pend* == `epptr()`

Referenced by std::wbuffer_convert< _Codecvt, _Elem, _Tr >::wbuffer_convert().

sgetc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sgetc ()  [inline], [inherited]
```

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.
Referenced by std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore(), and std::basic_istream< _CharT, _Traits >::ignore().

sgetn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sgetn (
            char_type * __s,
            streamsize __n)  [inline], [inherited]
```

Entry point for xsgetn.

Parameters

_↩ _s	A buffer area.
_↩ _n	A count.

Returns xsgetn(__s,__n). The effect is to fill __s[0] through __s[__n-1] with characters from the input sequence, if possible.

showmanyc()

```
template<typename _CharT, typename _Traits>
virtual streamsize std::basic_streambuf< _CharT, _Traits >::showmanyc ()  [inline], [protected],
[virtual], [inherited]
```

Investigating the data available.

Returns

An estimate of the number of characters available in the input sequence, or -1.

If it returns a positive value, then successive calls to `underflow()` *will not return* `traits::eof()` *until at least that number of characters have been supplied. If* `showmanyc()` *returns -1, then calls to* `underflow()` *or* `uflow()` *will fail.* [27.5.2.4.3]/1

Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to underflow or uflow] will not return* `eof()` *but that they will return immediately.*

The standard adds that *the morphemes of* `showmanyc` *are* ***es-how-many-see****, not* ***show-manic****.*

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< _CharT, std::char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< wchar_t >, std::basic_filebuf< wchar_t >, std::basic_stringbuf< _CharT, _Traits, _Alloc >, std::basic_stringbuf< char >, std::basic_stringbuf< char std::basic_stringbuf< wchar_t >, and std::basic_stringbuf< wchar_t >.

snextc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::snextc ()  [inline], [inherited]
```

Getting the next character.

Returns

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.
Referenced by std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< char_type, traits_type >::ignore(), std::basic_istream< char_type, traits_type >::operator>>(), and std::basic_istream< char_type, traits_type >::putback().

sputbackc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputbackc (
            char_type __c)  [inline], [inherited]
```

Pushing characters back into the input stream.

Parameters

_↩ _c	The character to push back.

Returns

The previous character, if possible.

Similar to sungetc(), but __c is pushed onto the stream instead of *the previous character.* If successful, the next character fetched from the input stream will be __c.
Referenced by std::basic_istream< _CharT, _Traits >::putback().

sputc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputc (
            char_type __c)  [inline], [inherited]
```

Entry point for all single-character output functions.

Parameters

_↩ _c	A character to output.

Returns

__c, if possible.

One of two public output functions.
If a write position is available for the output sequence (i.e., the buffer is not full), stores __c in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(↩ __c)`.
Referenced by std::wbuffer_convert< _Codecvt, _Elem, _Tr >::overflow().

sputn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sputn (
            const char_type * __s,
            streamsize __n)  [inline], [inherited]
```

Entry point for all single-character output functions.

Parameters

_↩ _s	A buffer read area.
_↩ _n	A count.

One of two public output functions.
Returns xsputn(__s,__n). The effect is to write __*s*[0] through __*s*[__n-1] to the output sequence, if possible.

state()

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Tr = char_traits<_Elem>>
state_type std::wbuffer_convert< _Codecvt, _Elem, _Tr >::state () const  [inline], [noexcept]
```
The conversion state following the last conversion.

sungetc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sungetc ()  [inline], [inherited]
```
Moving backwards in the input stream.

Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns pbackfail(). The effect is to *unget* the last character *gotten*.
Referenced by std::basic_istream< char_type, traits_type >::sentry::sentry().

sync()

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Tr = char_traits<_Elem>>
int std::wbuffer_convert< _Codecvt, _Elem, _Tr >::sync ()  [inline], [protected], [virtual]
```
Synchronizes the buffer arrays with the controlled sequences.

Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

Note

Base class version does nothing, returns zero.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

uflow()

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::uflow ()  [inline], [protected], [virtual],
[inherited]
```
Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.
Reimplemented in __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >.
Referenced by xsgetn().

underflow()

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Tr = char_traits<_Elem>>
_Wide_streambuf::int_type std::wbuffer_convert< _Codecvt, _Elem, _Tr >::underflow ()  [inline],
[protected], [virtual]
```

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.
For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.
A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References std::basic_streambuf< _CharT, _Traits >::egptr(), and std::basic_streambuf< _CharT, _Traits >::gptr().

xsgetn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::xsgetn (
            char_type * __s,
            streamsize __n)  [protected], [virtual], [inherited]
```

Multiple character extraction.

Parameters

_↩ _s	A buffer area.
_↩ _n	Maximum number of characters to assign.

Returns

The number of characters assigned.

Fills __s[0] through __s[__n-1] with characters from the input sequence, as if by `sbumpc()`. Stops when either __n characters have been copied, or when `traits::eof()` would be copied.
It is expected that derived classes provide a more efficient implementation by overriding this definition.
Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< _CharT, std::char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< char_type, traits_type >, and std::basic_filebuf< wchar_t >.
References egptr(), gptr(), std::min(), and uflow().

xsputn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::xsputn (
            const char_type * __s,
            streamsize __n)  [protected], [virtual], [inherited]
```

Multiple character insertion.

Parameters

_↩ _s	A buffer area.
_↩ _n	Maximum number of characters to write.

Returns

The number of characters written.

Writes __s[0] through __s[__n-1] to the output sequence, as if by `sputc()`. Stops when either *n* characters have been copied, or when `sputc()` would return `traits::eof()`.
It is expected that derived classes provide a more efficient implementation by overriding this definition.
Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< _CharT, std::char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< char_type, traits_type >, and std::basic_filebuf< wchar_t >.
References epptr(), std::min(), overflow(), and pptr().

5.1012.5 Member Data Documentation

_M_buf_locale

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::_M_buf_locale  [protected], [inherited]
```

Current locale setting.
Referenced by std::basic_filebuf< _CharT, _Traits >::basic_filebuf().

_M_in_beg

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_beg  [protected], [inherited]
```

Start of get area.

_M_in_cur

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_cur  [protected], [inherited]
```

Current read area.

_M_in_end

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_end  [protected], [inherited]
```

End of get area.

_M_out_beg

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_beg  [protected], [inherited]
```

Start of put area.

_M_out_cur

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_cur  [protected], [inherited]
```

Current put area.

_M_out_end

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_end  [protected], [inherited]
```

End of put area.

The documentation for this class was generated from the following file:

- locale_conv.h

5.1013 std::weak_ptr< _Tp > Class Template Reference

```
#include <memory>
```

Public Types

- using **element_type**

Public Member Functions

- template<typename _Yp, typename = _Constructible<const shared_ptr<_Yp>&>>
  weak_ptr (const shared_ptr< _Yp > &__r) noexcept
- **weak_ptr** (const weak_ptr &) noexcept=default
- template<typename _Yp, typename = _Constructible<const weak_ptr<_Yp>&>>
  weak_ptr (const weak_ptr< _Yp > &__r) noexcept
- **weak_ptr** (weak_ptr &&) noexcept=default
- template<typename _Yp, typename = _Constructible<weak_ptr<_Yp>>>
  weak_ptr (weak_ptr< _Yp > &&__r) noexcept
- bool **expired** () const noexcept
- shared_ptr< _Tp > **lock** () const noexcept
- template<typename _Yp>
  _Assignable< const shared_ptr< _Yp > & > **operator=** (const shared_ptr< _Yp > &__r) noexcept
- weak_ptr & **operator=** (const weak_ptr &__r) noexcept=default
- template<typename _Yp>
  _Assignable< const weak_ptr< _Yp > & > **operator=** (const weak_ptr< _Yp > &__r) noexcept
- weak_ptr & **operator=** (weak_ptr &&__r) noexcept=default
- template<typename _Yp>
  _Assignable< weak_ptr< _Yp > > **operator=** (weak_ptr< _Yp > &&__r) noexcept
- template<typename _Tp1>
  bool **owner_before** (const __shared_ptr< _Tp1, _Lp > &__rhs) const noexcept
- template<typename _Tp1>
  bool **owner_before** (const __weak_ptr< _Tp1, _Lp > &__rhs) const noexcept
- void **reset** () noexcept
- void **swap** (__weak_ptr &__s) noexcept
- long **use_count** () const noexcept

Related Symbols

(Note that these are not member symbols.)

- template<typename _Tp>
  void swap (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b) noexcept

5.1013.1 Detailed Description

template<typename _Tp>
class std::weak_ptr< _Tp >

A non-owning observer for a pointer owned by a shared_ptr.

Since

C++11

A weak_ptr provides a safe alternative to a raw pointer when you want a non-owning reference to an object that is managed by a shared_ptr.
Unlike a raw pointer, a weak_ptr can be converted to a new shared_ptr that shares ownership with every other shared_ptr that already owns the pointer. In other words you can upgrade from a non-owning "weak" reference to an owning shared_ptr, without having access to any of the existing shared_ptr objects.
Also unlike a raw pointer, a weak_ptr does not become "dangling" after the object it points to has been destroyed. Instead, a weak_ptr becomes *expired* and can no longer be converted to a shared_ptr that owns the freed pointer, so you cannot accidentally access the pointed-to object after it has been destroyed.
The documentation for this class was generated from the following file:

- bits/shared_ptr.h

5.1014 std::weibull_distribution< _RealType > Class Template Reference

`#include <random>`

Classes

- struct param_type

Public Types

- typedef _RealType result_type

Public Member Functions

- **weibull_distribution** (_RealType __a, _RealType __b=_RealType(1))
- **weibull_distribution** (const param_type &__p)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- _RealType a () const
- _RealType b () const
- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- bool operator== (const weibull_distribution &__d1, const weibull_distribution &__d2)

5.1014.1 Detailed Description

template<typename _RealType = double>
class std::weibull_distribution< _RealType >

A weibull_distribution random number distribution.
The formula for the normal probability density function is:

$$p(x|\alpha,\beta) = \frac{\alpha}{\beta}(\frac{x}{\beta})^{\alpha-1}\exp{(-(\frac{x}{\beta})^{\alpha})}$$

Since

C++11

5.1014.2 Member Typedef Documentation

result_type

```
template<typename _RealType = double>
typedef _RealType std::weibull_distribution< _RealType >::result_type
```

The type of the range of the distribution.

5.1014.3 Member Function Documentation

a()

```
template<typename _RealType = double>
_RealType std::weibull_distribution< _RealType >::a () const  [inline]
```

Return the a parameter of the distribution.

b()

```
template<typename _RealType = double>
_RealType std::weibull_distribution< _RealType >::b () const  [inline]
```

Return the b parameter of the distribution.

max()

```
template<typename _RealType = double>
result_type std::weibull_distribution< _RealType >::max () const  [inline]
```

Returns the least upper bound value of the distribution.

min()

```
template<typename _RealType = double>
result_type std::weibull_distribution< _RealType >::min () const  [inline]
```

Returns the greatest lower bound value of the distribution.

operator()()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
```

```
result_type std::weibull_distribution< _RealType >::operator() (
            _UniformRandomNumberGenerator & __urng)  [inline]
```

Generating functions.
Referenced by operator()().

param() [1/2]

```
template<typename _RealType = double>
param_type std::weibull_distribution< _RealType >::param () const  [inline]
```

Returns the parameter set of the distribution.
Referenced by std::operator>>().

param() [2/2]

```
template<typename _RealType = double>
void std::weibull_distribution< _RealType >::param (
            const param_type & __param)  [inline]
```

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

reset()

```
template<typename _RealType = double>
void std::weibull_distribution< _RealType >::reset ()  [inline]
```

Resets the distribution state.

5.1014.4 Friends And Related Symbol Documentation

operator==

```
template<typename _RealType = double>
bool operator== (
            const weibull_distribution< _RealType > & __d1,
            const weibull_distribution< _RealType > & __d2)  [friend]
```

Return true if two Weibull distributions have the same parameters.
The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

5.1015 std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc > Class Template Reference

```
#include <locale_conv.h>
```

Public Types

- typedef basic_string< char, char_traits< char >, _Byte_alloc > **byte_string**
- typedef wide_string::traits_type::int_type **int_type**
- typedef _Codecvt::state_type **state_type**
- typedef basic_string< _Elem, char_traits< _Elem >, _Wide_alloc > **wide_string**

Public Member Functions

- wstring_convert ()
- wstring_convert (_Codecvt *__pcvt)
- wstring_convert (_Codecvt *__pcvt, state_type __state)
- wstring_convert (const byte_string &__byte_err, const wide_string &__wide_err=wide_string())
- **wstring_convert** (const wstring_convert &)=delete
- size_t converted () const noexcept
- wstring_convert & **operator=** (const wstring_convert &)=delete
- state_type state () const

- wide_string from_bytes (char __byte)
- wide_string from_bytes (const char *__ptr)
- wide_string from_bytes (const byte_string &__str)
- wide_string from_bytes (const char *__first, const char *__last)

- byte_string to_bytes (_Elem __wchar)
- byte_string to_bytes (const _Elem *__ptr)
- byte_string to_bytes (const wide_string &__wstr)
- byte_string to_bytes (const _Elem *__first, const _Elem *__last)

5.1015.1 Detailed Description

template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>, typename _Byte_alloc = allocator<char>>
class std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >

String conversions.

5.1015.2 Constructor & Destructor Documentation

wstring_convert() [1/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::wstring_convert ()  [inline]
```

Default constructor.

wstring_convert() [2/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::wstring_convert (
            _Codecvt * __pcvt)  [inline], [explicit]
```

Constructor.

Parameters

__pcvt	The facet to use for conversions.

Takes ownership of __pcvt and will delete it in the destructor.

wstring_convert() [3/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::wstring_convert (
            _Codecvt * __pcvt,
            state_type __state)  [inline]
```

Construct with an initial converstion state.

Parameters

__pcvt	The facet to use for conversions.
__state	Initial conversion state.

Takes ownership of __pcvt and will delete it in the destructor. The object's conversion state will persist between conversions.

wstring_convert() [4/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::wstring_convert (
            const byte_string & __byte_err,
            const wide_string & __wide_err = wide_string())  [inline], [explicit]
```

Construct with error strings.

Parameters

__byte_err	A string to return on failed conversions.
__wide_err	A wide string to return on failed conversions.

5.1015.3 Member Function Documentation

converted()

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
size_t std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::converted () const
[inline], [noexcept]
```

The number of elements successfully converted in the last conversion.

from_bytes() [1/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
wide_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::from_bytes (
            char __byte)  [inline]
```

Convert from bytes.
References from_bytes().
Referenced by from_bytes(), from_bytes(), and from_bytes().

from_bytes() [2/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
wide_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::from_bytes (
            const byte_string & __str)  [inline]
```

Convert from bytes.
References std::basic_string< _CharT, _Traits, _Alloc >::data(), from_bytes(), and std::basic_string< _CharT, _Traits, _Alloc >::size().

from_bytes() [3/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
wide_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::from_bytes (
            const char * __first,
            const char * __last)  [inline]
```

Convert from bytes.

from_bytes() [4/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
wide_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::from_bytes (
            const char * __ptr)  [inline]
```

Convert from bytes.
References from_bytes().

state()

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
state_type std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::state () const
[inline]
```

The final conversion state of the last conversion.

to_bytes() [1/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
byte_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::to_bytes (
            _Elem __wchar)  [inline]
```

Convert to bytes.
References to_bytes().
Referenced by to_bytes(), to_bytes(), and to_bytes().

to_bytes() [2/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
byte_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::to_bytes (
            const _Elem * __first,
            const _Elem * __last)  [inline]
```

Convert to bytes.

to_bytes() [3/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
byte_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::to_bytes (
            const _Elem * __ptr)  [inline]
```

Convert to bytes.
References std::basic_string< _Elem, char_traits< _Elem >, _Wide_alloc >::length(), and to_bytes().

to_bytes() [4/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
byte_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::to_bytes (
            const wide_string & __wstr)  [inline]
```

Convert to bytes.
References std::basic_string< _CharT, _Traits, _Alloc >::data(), std::basic_string< _CharT, _Traits, _Alloc >::size(), and to_bytes().
The documentation for this class was generated from the following file:

- locale_conv.h

6 File Documentation

6.1 algorithm File Reference

Macros

- #define **__glibcxx_want_algorithm_default_value_type**
- #define **__glibcxx_want_clamp**
- #define **__glibcxx_want_constexpr_algorithms**
- #define **__glibcxx_want_freestanding_algorithm**
- #define **__glibcxx_want_parallel_algorithm**
- #define **__glibcxx_want_ranges**
- #define **__glibcxx_want_ranges_contains**
- #define **__glibcxx_want_ranges_find_last**
- #define **__glibcxx_want_ranges_fold**
- #define **__glibcxx_want_ranges_starts_ends_with**
- #define **__glibcxx_want_robust_nonmodifying_seq_ops**
- #define **__glibcxx_want_sample**
- #define **__glibcxx_want_shift**
- #define **_GLIBCXX_ALGORITHM**

6.1.1 Detailed Description

This is a Standard C++ Library header.

6.2 experimental/algorithm File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **__cpp_lib_experimental_sample**
- #define **_GLIBCXX_EXPERIMENTAL_ALGORITHM**

Functions

- template<typename _PopulationIterator, typename _SampleIterator, typename _Distance>
  _SampleIterator **std::experimental::sample** (_PopulationIterator __first, _PopulationIterator __last, _SampleIterator __out, _Distance __n)
- template<typename _PopulationIterator, typename _SampleIterator, typename _Distance, typename _UniformRandomNumberGenerator>
  _SampleIterator std::experimental::sample (_PopulationIterator __first, _PopulationIterator __last, _SampleIterator __out, _Distance __n, _UniformRandomNumberGenerator &&__g)
- template<typename _ForwardIterator, typename _Searcher>
  _ForwardIterator **std::experimental::search** (_ForwardIterator __first, _ForwardIterator __last, const _Searcher &__searcher)
- template<typename _RandomAccessIterator>
  void **std::experimental::shuffle** (_RandomAccessIterator __first, _RandomAccessIterator __last)

6.2.1 Detailed Description

This is a TS C++ Library header.

6.3 ext/algorithm File Reference

Namespaces

- namespace __gnu_cxx

Macros

- #define **_EXT_ALGORITHM**

Functions

- template<typename _InputIterator, typename _Size, typename _OutputIterator>
  std::pair< _InputIterator, _OutputIterator > **__gnu_cxx::__copy_n** (_InputIterator __first, _Size __count, _OutputIterator __result, std::input_iterator_tag)
- template<typename _RAIterator, typename _Size, typename _OutputIterator>
  std::pair< _RAIterator, _OutputIterator > **__gnu_cxx::__copy_n** (_RAIterator __first, _Size __count, _OutputIterator __result, std::random_access_iterator_tag)
- template<typename _InputIterator1, typename _InputIterator2>
  int **__gnu_cxx::__lexicographical_compare_3way** (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)
- int **__gnu_cxx::__lexicographical_compare_3way** (const char *__first1, const char *__last1, const char *_first2, const char *__last2)
- int **__gnu_cxx::__lexicographical_compare_3way** (const unsigned char *__first1, const unsigned char *__last1, const unsigned char *__first2, const unsigned char *__last2)
- template<typename _Tp>
  const _Tp & __gnu_cxx::__median (const _Tp &__a, const _Tp &__b, const _Tp &__c)
- template<typename _Tp, typename _Compare>
  const _Tp & __gnu_cxx::__median (const _Tp &__a, const _Tp &__b, const _Tp &__c, _Compare __comp)
- template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator, typename _Distance>
  _RandomAccessIterator **__gnu_cxx::__random_sample** (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out, _RandomNumberGenerator &__rand, const _Distance __n)

- template<typename _InputIterator, typename _RandomAccessIterator, typename _Distance>
  _RandomAccessIterator **__gnu_cxx::__random_sample** (_InputIterator __first, _InputIterator __last, _↩
  RandomAccessIterator __out, const _Distance __n)
- template<typename _InputIterator, typename _Size, typename _OutputIterator>
  std::pair< _InputIterator, _OutputIterator > __gnu_cxx::copy_n (_InputIterator __first, _Size __count, _Output↩
  Iterator __result)
- template<typename _InputIterator, typename _Tp, typename _Size>
  void **__gnu_cxx::count** (_InputIterator __first, _InputIterator __last, const _Tp &__value, _Size &__n)
- template<typename _InputIterator, typename _Predicate, typename _Size>
  void **__gnu_cxx::count_if** (_InputIterator __first, _InputIterator __last, _Predicate __pred, _Size &__n)
- template<typename _RAIter>
  constexpr bool **__gnu_cxx::is_heap** (_RAIter, _RAIter)
- template<typename _FIter>
  constexpr bool **__gnu_cxx::is_sorted** (_FIter, _FIter)
- template<typename _InputIterator1, typename _InputIterator2>
  int __gnu_cxx::lexicographical_compare_3way (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)
- template<typename _InputIterator, typename _RandomAccessIterator>
  _RandomAccessIterator __gnu_cxx::random_sample (_InputIterator __first, _InputIterator __last, _Random↩
  AccessIterator __out_first, _RandomAccessIterator __out_last)
- template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator>
  _RandomAccessIterator __gnu_cxx::random_sample (_InputIterator __first, _InputIterator __last, _Random↩
  AccessIterator __out_first, _RandomAccessIterator __out_last, _RandomNumberGenerator &__rand)
- template<typename _ForwardIterator, typename _OutputIterator, typename _Distance>
  _OutputIterator __gnu_cxx::random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const _Distance __n)
- template<typename _ForwardIterator, typename _OutputIterator, typename _Distance, typename _RandomNumberGenerator>
  _OutputIterator __gnu_cxx::random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const _Distance __n, _RandomNumberGenerator &__rand)

6.3.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

6.4 parallel/algorithm File Reference

Macros

- #define **_PARALLEL_ALGORITHM**

6.4.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

6.5 any File Reference

Macros

- #define **__glibcxx_want_any**
- #define **_GLIBCXX_ANY**

6.5.1 Detailed Description

This is a Standard C++ Library header.

6.6 experimental/any File Reference

Classes

- class std::experimental::fundamentals_v1::any
- class std::experimental::fundamentals_v1::bad_any_cast

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **__cpp_lib_experimental_any**
- #define **_GLIBCXX_EXPERIMENTAL_ANY**

Functions

- template<typename _ValueType>
  _ValueType std::experimental::any_cast (const any &__any)
- void std::experimental::swap (any &__x, any &__y) noexcept

- template<typename _ValueType, typename enable_if<!is_move_constructible< _ValueType >::value||is_lvalue_reference< _ValueType >::value, bool >::type = true>
  _ValueType std::experimental::any_cast (any &&__any)
- template<typename _ValueType, typename enable_if< is_move_constructible< _ValueType >::value &&!is_lvalue_reference< _ValueType >::value, bool >::type = false>
  _ValueType std::experimental::any_cast (any &&__any)
- template<typename _ValueType>
  _ValueType std::experimental::any_cast (any &__any)

- template<typename _ValueType>
  _ValueType * std::experimental::any_cast (any *__any) noexcept
- template<typename _ValueType>
  const _ValueType * std::experimental::any_cast (const any *__any) noexcept

6.6.1 Detailed Description

This is a TS C++ Library header.

6.7 array File Reference

Classes

- struct std::array< _Tp, _Nm >
- struct std::tuple_element< _Ind, array< _Tp, _Nm > >
- struct std::tuple_size< array< _Tp, _Nm > >

Namespaces

- namespace std

Macros

- #define **__glibcxx_want_array_constexpr**
- #define **__glibcxx_want_freestanding_array**
- #define **__glibcxx_want_nonmember_container_access**
- #define **__glibcxx_want_to_array**
- #define **_GLIBCXX_ARRAY**

Functions

- template<typename _Tp, typename... _Up>
  std::array (_Tp, _Up...) -> array< enable_if_t<(is_same_v< _Tp, _Up > &&...), _Tp >, 1+sizeof...(_Up)>
- template<std::size_t _Int, typename _Tp, std::size_t _Nm>
  constexpr _Tp && **std::get** (array< _Tp, _Nm > &&__arr) noexcept
- template<std::size_t _Int, typename _Tp, std::size_t _Nm>
  constexpr _Tp & **std::get** (array< _Tp, _Nm > &__arr) noexcept
- template<std::size_t _Int, typename _Tp, std::size_t _Nm>
  constexpr const _Tp && **std::get** (const array< _Tp, _Nm > &&__arr) noexcept
- template<std::size_t _Int, typename _Tp, std::size_t _Nm>
  constexpr const _Tp & **std::get** (const array< _Tp, _Nm > &__arr) noexcept
- template<typename _Tp, size_t _Nm>
  constexpr __detail::__synth3way_t< _Tp > **std::operator<=>** (const array< _Tp, _Nm > &__a, const array< _Tp, _Nm > &__b)
- template<typename _Tp, std::size_t _Nm>
  constexpr bool **std::operator==** (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)
- template<typename _Tp, std::size_t _Nm>
  __enable_if_t<!__array_traits< _Tp, _Nm >::_Is_swappable::value > **std::swap** (array< _Tp, _Nm > &, array< _Tp, _Nm > &)=delete
- template<typename _Tp, std::size_t _Nm>
  constexpr __enable_if_t< __array_traits< _Tp, _Nm >::_Is_swappable::value > **std::swap** (array< _Tp, _Nm > &__one, array< _Tp, _Nm > &__two) noexcept(noexcept(__one.swap(__two)))

Variables

- template<typename _Tp, size_t _Nm>
  constexpr size_t **std::tuple_size_v**< **array**< **_Tp, _Nm** > >
- template<typename _Tp, size_t _Nm>
  constexpr size_t **std::tuple_size_v**< **const array**< **_Tp, _Nm** > >

6.7.1 Detailed Description

This is a Standard C++ Library header.

6.8 experimental/array File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **__cpp_lib_experimental_make_array**
- #define **_GLIBCXX_EXPERIMENTAL_ARRAY**

Functions

- template<typename _Tp, size_t _Nm, size_t... _Idx>
  constexpr array< remove_cv_t< _Tp >, _Nm > **std::experimental::__to_array** (_Tp(&__a)[_Nm], index_sequence< _Idx... >)
- template<typename _Dest = void, typename... _Types>
  constexpr array< typename __make_array_elem< _Dest, _Types... >::type, sizeof...(_Types)> std::experimental::make_array (_Types &&... __t)
- template<typename _Tp, size_t _Nm>
  constexpr array< remove_cv_t< _Tp >, _Nm > std::experimental::to_array (_Tp(&__a)[_Nm]) noexcept(is_nothrow_constructible< remove_cv_t< _Tp >, _Tp & >::value)

6.8.1 Detailed Description

This is a TS C++ Library header.

6.9 atomic File Reference

Classes

- struct std::atomic< _Tp >
- struct std::atomic< _Tp * >

Namespaces

- namespace std

Macros

- #define **__glibcxx_want_atomic_flag_test**
- #define **__glibcxx_want_atomic_float**
- #define **__glibcxx_want_atomic_is_always_lock_free**
- #define **__glibcxx_want_atomic_lock_free_type_aliases**
- #define **__glibcxx_want_atomic_ref**
- #define **__glibcxx_want_atomic_value_initialization**
- #define **__glibcxx_want_atomic_wait**
- #define **_GLIBCXX_ATOMIC**

Typedefs

- typedef atomic< bool > std::atomic_bool
- typedef atomic< char > std::atomic_char
- typedef atomic< char16_t > std::atomic_char16_t
- typedef atomic< char32_t > std::atomic_char32_t
- typedef atomic< int > std::atomic_int
- typedef atomic< int16_t > std::atomic_int16_t
- typedef atomic< int32_t > std::atomic_int32_t
- typedef atomic< int64_t > std::atomic_int64_t
- typedef atomic< int8_t > std::atomic_int8_t
- typedef atomic< int_fast16_t > std::atomic_int_fast16_t
- typedef atomic< int_fast32_t > std::atomic_int_fast32_t
- typedef atomic< int_fast64_t > std::atomic_int_fast64_t
- typedef atomic< int_fast8_t > std::atomic_int_fast8_t
- typedef atomic< int_least16_t > std::atomic_int_least16_t

- typedef atomic< int_least32_t > std::atomic_int_least32_t
- typedef atomic< int_least64_t > std::atomic_int_least64_t
- typedef atomic< int_least8_t > std::atomic_int_least8_t
- typedef atomic< intmax_t > std::atomic_intmax_t
- typedef atomic< intptr_t > std::atomic_intptr_t
- typedef atomic< long long > std::atomic_llong
- typedef atomic< long > std::atomic_long
- typedef atomic< ptrdiff_t > std::atomic_ptrdiff_t
- typedef atomic< signed char > std::atomic_schar
- typedef atomic< short > std::atomic_short
- typedef atomic< size_t > std::atomic_size_t
- typedef atomic< unsigned char > std::atomic_uchar
- typedef atomic< unsigned int > std::atomic_uint
- typedef atomic< uint16_t > std::atomic_uint16_t
- typedef atomic< uint32_t > std::atomic_uint32_t
- typedef atomic< uint64_t > std::atomic_uint64_t
- typedef atomic< uint8_t > std::atomic_uint8_t
- typedef atomic< uint_fast16_t > std::atomic_uint_fast16_t
- typedef atomic< uint_fast32_t > std::atomic_uint_fast32_t
- typedef atomic< uint_fast64_t > std::atomic_uint_fast64_t
- typedef atomic< uint_fast8_t > std::atomic_uint_fast8_t
- typedef atomic< uint_least16_t > std::atomic_uint_least16_t
- typedef atomic< uint_least32_t > std::atomic_uint_least32_t
- typedef atomic< uint_least64_t > std::atomic_uint_least64_t
- typedef atomic< uint_least8_t > std::atomic_uint_least8_t
- typedef atomic< uintmax_t > std::atomic_uintmax_t
- typedef atomic< uintptr_t > std::atomic_uintptr_t
- typedef atomic< unsigned long long > std::atomic_ullong
- typedef atomic< unsigned long > std::atomic_ulong
- typedef atomic< unsigned short > std::atomic_ushort
- typedef atomic< wchar_t > std::atomic_wchar_t

Functions

- template<typename _ITp>
  bool **std::atomic_compare_exchange_strong** (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __↩
  atomic_val_t< _ITp > __i2) noexcept
- template<typename _ITp>
  bool **std::atomic_compare_exchange_strong** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1,
  __atomic_val_t< _ITp > __i2) noexcept
- template<typename _ITp>
  bool **std::atomic_compare_exchange_strong_explicit** (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1,
  __atomic_val_t< _ITp > __i2, memory_order __m1, memory_order __m2) noexcept
- template<typename _ITp>
  bool **std::atomic_compare_exchange_strong_explicit** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp
  > *__i1, __atomic_val_t< _ITp > __i2, memory_order __m1, memory_order __m2) noexcept
- template<typename _ITp>
  bool **std::atomic_compare_exchange_weak** (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __↩
  atomic_val_t< _ITp > __i2) noexcept
- template<typename _ITp>
  bool **std::atomic_compare_exchange_weak** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1,
  __atomic_val_t< _ITp > __i2) noexcept

- template<typename _ITp>
  bool **std::atomic_compare_exchange_weak_explicit** (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2, memory_order __m1, memory_order __m2) noexcept
- template<typename _ITp>
  bool **std::atomic_compare_exchange_weak_explicit** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2, memory_order __m1, memory_order __m2) noexcept
- template<typename _ITp>
  _ITp **std::atomic_exchange** (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **std::atomic_exchange** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **std::atomic_exchange_explicit** (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **std::atomic_exchange_explicit** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_add** (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_add** (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_add_explicit** (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_add_explicit** (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_and** (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_and** (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_and_explicit** (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_and_explicit** (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_or** (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_or** (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_or_explicit** (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_or_explicit** (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_sub** (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_sub** (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_sub_explicit** (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order __m) noexcept

- template<typename _ITp>
  _ITp **std::atomic_fetch_sub_explicit** (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_xor** (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_xor** (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_xor_explicit** (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_xor_explicit** (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- void **std::atomic_flag_clear** (atomic_flag *__a) noexcept
- void **std::atomic_flag_clear** (volatile atomic_flag *__a) noexcept
- void **std::atomic_flag_clear_explicit** (atomic_flag *__a, memory_order __m) noexcept
- void **std::atomic_flag_clear_explicit** (volatile atomic_flag *__a, memory_order __m) noexcept
- bool **std::atomic_flag_test_and_set** (atomic_flag *__a) noexcept
- bool **std::atomic_flag_test_and_set** (volatile atomic_flag *__a) noexcept
- bool **std::atomic_flag_test_and_set_explicit** (atomic_flag *__a, memory_order __m) noexcept
- bool **std::atomic_flag_test_and_set_explicit** (volatile atomic_flag *__a, memory_order __m) noexcept
- template<typename _ITp>
  void **std::atomic_init** (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  void **std::atomic_init** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  bool **std::atomic_is_lock_free** (const atomic< _ITp > *__a) noexcept
- template<typename _ITp>
  bool **std::atomic_is_lock_free** (const volatile atomic< _ITp > *__a) noexcept
- template<typename _ITp>
  _ITp **std::atomic_load** (const atomic< _ITp > *__a) noexcept
- template<typename _ITp>
  _ITp **std::atomic_load** (const volatile atomic< _ITp > *__a) noexcept
- template<typename _ITp>
  _ITp **std::atomic_load_explicit** (const atomic< _ITp > *__a, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **std::atomic_load_explicit** (const volatile atomic< _ITp > *__a, memory_order __m) noexcept
- template<typename _ITp>
  void **std::atomic_store** (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  void **std::atomic_store** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  void **std::atomic_store_explicit** (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  void **std::atomic_store_explicit** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept

6.9.1 Detailed Description

This is a Standard C++ Library header.

6.10 auto_ptr.h File Reference

Classes

- class std::auto_ptr< _Tp >
- struct std::auto_ptr_ref< _Tp1 >

Namespaces

- namespace std

6.10.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

6.11 backward_warning.h File Reference

6.11.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iosfwd>.

6.12 hash_fun.h File Reference

Namespaces

- namespace __gnu_cxx

Typedefs

- typedef __SIZE_TYPE__ **__gnu_cxx::size_t**

Functions

- size_t **__gnu_cxx::__stl_hash_string** (const char *__s)

6.12.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

6.13 hash_map File Reference

Classes

- class __gnu_cxx::allocator< _Tp >
- struct __gnu_cxx::equal_to< _Tp >
- class __gnu_cxx::hash_map< _Key, _Tp, _HashFn, _EqualKey, _Alloc >
- class __gnu_cxx::hash_multimap< _Key, _Tp, _HashFn, _EqualKey, _Alloc >
- struct __gnu_cxx::pair< _T1, _T2 >

Namespaces

- namespace __gnu_cxx
- namespace std

Macros

- #define **_BACKWARD_HASH_MAP**

Functions

- template<class _Key, class _Tp, class _HashFn, class _EqlKey, class _Alloc>
  bool **__gnu_cxx::operator!=** (const hash_map< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm1, const hash_map< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm2)
- template<class _Key, class _Tp, class _HF, class _EqKey, class _Alloc>
  bool **__gnu_cxx::operator!=** (const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm1, const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm2)
- template<class _Key, class _Tp, class _HashFn, class _EqlKey, class _Alloc>
  bool **__gnu_cxx::operator==** (const hash_map< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm1, const hash_map< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm2)
- template<class _Key, class _Tp, class _HF, class _EqKey, class _Alloc>
  bool **__gnu_cxx::operator==** (const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm1, const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm2)
- template<class _Key, class _Tp, class _HashFn, class _EqlKey, class _Alloc>
  void **__gnu_cxx::swap** (hash_map< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm1, hash_map< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm2)
- template<class _Key, class _Tp, class _HashFn, class _EqlKey, class _Alloc>
  void **__gnu_cxx::swap** (hash_multimap< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm1, hash_multimap< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm2)

6.13.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

6.14 hash_set File Reference

Classes

- class __gnu_cxx::hash_multiset< _Value, _HashFcn, _EqualKey, _Alloc >
- class __gnu_cxx::hash_set< _Value, _HashFcn, _EqualKey, _Alloc >

Namespaces

- namespace __gnu_cxx
- namespace std

Macros

- #define **_BACKWARD_HASH_SET**

Functions

- template<class _Val, class _HashFcn, class _EqualKey, class _Alloc>
  bool **__gnu_cxx::operator!=** (const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)
- template<class _Value, class _HashFcn, class _EqualKey, class _Alloc>
  bool **__gnu_cxx::operator!=** (const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs2)

- template<class _Val, class _HashFcn, class _EqualKey, class _Alloc>
  bool **__gnu_cxx::operator==** (const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)
- template<class _Value, class _HashFcn, class _EqualKey, class _Alloc>
  bool **__gnu_cxx::operator==** (const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs2)
- template<class _Val, class _HashFcn, class _EqualKey, class _Alloc>
  void **__gnu_cxx::swap** (hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)
- template<class _Val, class _HashFcn, class _EqualKey, class _Alloc>
  void **__gnu_cxx::swap** (hash_set< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, hash_set< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)

6.14.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

6.15 strstream File Reference

Namespaces

- namespace std

Macros

- #define **_BACKWARD_STRSTREAM**
- #define **_GLIBCXX_STRSTREAM_DEPR**(A, B)

6.15.1 Detailed Description

This is a Standard C++ Library header.

6.16 barrier File Reference

Macros

- #define **__glibcxx_want_barrier**
- #define **_GLIBCXX_BARRIER**

6.16.1 Detailed Description

This is a Standard C++ Library header.

6.17 bit File Reference

Namespaces

- namespace std

Macros

- #define **__glibcxx_want_bit_cast**
- #define **__glibcxx_want_bitops**
- #define **__glibcxx_want_byteswap**
- #define **__glibcxx_want_endian**
- #define **__glibcxx_want_int_pow2**
- #define **_GLIBCXX_BIT**

6.17.1 Detailed Description

This is a Standard C++ Library header.

6.18 bits/algorithmfwd.h File Reference

Namespaces

- namespace std

Functions

- template<typename _FIter>
  constexpr _FIter **std::adjacent_find** (_FIter, _FIter)
- template<typename _FIter, typename _BinaryPredicate>
  constexpr _FIter **std::adjacent_find** (_FIter, _FIter, _BinaryPredicate)
- template<typename _IIter, typename _Predicate>
  constexpr bool **std::all_of** (_IIter, _IIter, _Predicate)
- template<typename _IIter, typename _Predicate>
  constexpr bool **std::any_of** (_IIter, _IIter, _Predicate)
- template<typename _FIter, typename _Tp>
  constexpr bool **std::binary_search** (_FIter, _FIter, const _Tp &)
- template<typename _FIter, typename _Tp, typename _Compare>
  constexpr bool **std::binary_search** (_FIter, _FIter, const _Tp &, _Compare)
- template<typename _Tp>
  constexpr const _Tp & std::clamp (const _Tp &__val, const _Tp &__lo, const _Tp &__hi)
- template<typename _Tp, typename _Compare>
  constexpr const _Tp & std::clamp (const _Tp &__val, const _Tp &__lo, const _Tp &__hi, _Compare __comp)
- template<typename _IIter, typename _OIter>
  constexpr _OIter **std::copy** (_IIter, _IIter, _OIter)
- template<typename _BIter1, typename _BIter2>
  constexpr _BIter2 **std::copy_backward** (_BIter1, _BIter1, _BIter2)
- template<typename _IIter, typename _OIter, typename _Predicate>
  constexpr _OIter **std::copy_if** (_IIter, _IIter, _OIter, _Predicate)
- template<typename _IIter, typename _Size, typename _OIter>
  constexpr _OIter **std::copy_n** (_IIter, _Size, _OIter)
- template<typename _IIter, typename _Tp>
  constexpr iterator_traits< _IIter >::difference_type **std::count** (_IIter, _IIter, const _Tp &)
- template<typename _IIter, typename _Predicate>
  constexpr iterator_traits< _IIter >::difference_type **std::count_if** (_IIter, _IIter, _Predicate)
- template<typename _IIter1, typename _IIter2, typename _BinaryPredicate>
  constexpr bool std::equal (_IIter1 __first1, _IIter1 __last1, _IIter2 __first2, _BinaryPredicate __binary_pred)
- template<typename _IIter1, typename _IIter2>
  constexpr bool **std::equal** (_IIter1, _IIter1, _IIter2)
- template<typename _FIter, typename _Tp>
  constexpr pair< _FIter, _FIter > **std::equal_range** (_FIter, _FIter, const _Tp &)
- template<typename _FIter, typename _Tp, typename _Compare>
  constexpr pair< _FIter, _FIter > **std::equal_range** (_FIter, _FIter, const _Tp &, _Compare)
- template<typename _FIter, typename _Tp>
  constexpr void **std::fill** (_FIter, _FIter, const _Tp &)
- template<typename _OIter, typename _Size, typename _Tp>
  constexpr _OIter **std::fill_n** (_OIter, _Size, const _Tp &)
- template<typename _IIter, typename _Tp>
  constexpr _IIter **std::find** (_IIter, _IIter, const _Tp &)

- template<typename _FIter1, typename _FIter2>
  constexpr _FIter1 **std::find_end** (_FIter1, _FIter1, _FIter2, _FIter2)
- template<typename _FIter1, typename _FIter2, typename _BinaryPredicate>
  constexpr _FIter1 **std::find_end** (_FIter1, _FIter1, _FIter2, _FIter2, _BinaryPredicate)
- template<typename _FIter1, typename _FIter2>
  constexpr _FIter1 **std::find_first_of** (_FIter1, _FIter1, _FIter2, _FIter2)
- template<typename _FIter1, typename _FIter2, typename _BinaryPredicate>
  constexpr _FIter1 **std::find_first_of** (_FIter1, _FIter1, _FIter2, _FIter2, _BinaryPredicate)
- template<typename _IIter, typename _Predicate>
  constexpr _IIter **std::find_if** (_IIter, _IIter, _Predicate)
- template<typename _IIter, typename _Predicate>
  constexpr _IIter **std::find_if_not** (_IIter, _IIter, _Predicate)
- template<typename _IIter, typename _Funct>
  constexpr _Funct **std::for_each** (_IIter, _IIter, _Funct)
- template<typename _FIter, typename _Generator>
  constexpr void **std::generate** (_FIter, _FIter, _Generator)
- template<typename _OIter, typename _Size, typename _Generator>
  constexpr _OIter **std::generate_n** (_OIter, _Size, _Generator)
- template<typename _IIter1, typename _IIter2>
  constexpr bool **std::includes** (_IIter1, _IIter1, _IIter2, _IIter2)
- template<typename _IIter1, typename _IIter2, typename _Compare>
  constexpr bool **std::includes** (_IIter1, _IIter1, _IIter2, _IIter2, _Compare)
- template<typename _BIter>
  void **std::inplace_merge** (_BIter, _BIter, _BIter)
- template<typename _BIter, typename _Compare>
  void **std::inplace_merge** (_BIter, _BIter, _BIter, _Compare)
- template<typename _RAIter>
  constexpr bool **std::is_heap** (_RAIter, _RAIter)
- template<typename _RAIter, typename _Compare>
  constexpr bool **std::is_heap** (_RAIter, _RAIter, _Compare)
- template<typename _RAIter>
  constexpr _RAIter **std::is_heap_until** (_RAIter, _RAIter)
- template<typename _RAIter, typename _Compare>
  constexpr _RAIter **std::is_heap_until** (_RAIter, _RAIter, _Compare)
- template<typename _IIter, typename _Predicate>
  constexpr bool **std::is_partitioned** (_IIter, _IIter, _Predicate)
- template<typename _FIter1, typename _FIter2>
  constexpr bool **std::is_permutation** (_FIter1, _FIter1, _FIter2)
- template<typename _FIter1, typename _FIter2, typename _BinaryPredicate>
  constexpr bool **std::is_permutation** (_FIter1, _FIter1, _FIter2, _BinaryPredicate)
- template<typename _FIter>
  constexpr bool **std::is_sorted** (_FIter, _FIter)
- template<typename _FIter, typename _Compare>
  constexpr bool **std::is_sorted** (_FIter, _FIter, _Compare)
- template<typename _FIter>
  constexpr _FIter **std::is_sorted_until** (_FIter, _FIter)
- template<typename _FIter, typename _Compare>
  constexpr _FIter **std::is_sorted_until** (_FIter, _FIter, _Compare)
- template<typename _FIter1, typename _FIter2>
  constexpr void **std::iter_swap** (_FIter1, _FIter2)
- template<typename _IIter1, typename _IIter2>
  constexpr bool **std::lexicographical_compare** (_IIter1, _IIter1, _IIter2, _IIter2)

- template<typename _IIter1, typename _IIter2, typename _Compare>
  constexpr bool **std::lexicographical_compare** (_IIter1, _IIter1, _IIter2, _IIter2, _Compare)
- template<typename _FIter, typename _Tp>
  constexpr _FIter **std::lower_bound** (_FIter, _FIter, const _Tp &)
- template<typename _FIter, typename _Tp, typename _Compare>
  constexpr _FIter **std::lower_bound** (_FIter, _FIter, const _Tp &, _Compare)
- template<typename _RAIter>
  constexpr void **std::make_heap** (_RAIter, _RAIter)
- template<typename _RAIter, typename _Compare>
  constexpr void **std::make_heap** (_RAIter, _RAIter, _Compare)
- template<typename _Tp>
  constexpr const _Tp & std::max (const _Tp &__a, const _Tp &__b)
- template<typename _Tp, typename _Compare>
  constexpr const _Tp & std::max (const _Tp &__a, const _Tp &__b, _Compare __comp)
- template<typename _Tp>
  constexpr _Tp **std::max** (initializer_list< _Tp >)
- template<typename _Tp, typename _Compare>
  constexpr _Tp **std::max** (initializer_list< _Tp >, _Compare)
- template<typename _FIter>
  constexpr _FIter **std::max_element** (_FIter, _FIter)
- template<typename _FIter, typename _Compare>
  constexpr _FIter **std::max_element** (_FIter, _FIter, _Compare)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  constexpr _OIter **std::merge** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare>
  constexpr _OIter **std::merge** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)
- template<typename _Tp>
  constexpr const _Tp & std::min (const _Tp &__a, const _Tp &__b)
- template<typename _Tp, typename _Compare>
  constexpr const _Tp & std::min (const _Tp &__a, const _Tp &__b, _Compare __comp)
- template<typename _Tp>
  constexpr _Tp **std::min** (initializer_list< _Tp >)
- template<typename _Tp, typename _Compare>
  constexpr _Tp **std::min** (initializer_list< _Tp >, _Compare)
- template<typename _FIter>
  constexpr _FIter **std::min_element** (_FIter, _FIter)
- template<typename _FIter, typename _Compare>
  constexpr _FIter **std::min_element** (_FIter, _FIter, _Compare)
- template<typename _Tp>
  constexpr pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b)
- template<typename _Tp, typename _Compare>
  constexpr pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b, _Compare __comp)
- template<typename _Tp>
  constexpr pair< _Tp, _Tp > **std::minmax** (initializer_list< _Tp >)
- template<typename _Tp, typename _Compare>
  constexpr pair< _Tp, _Tp > **std::minmax** (initializer_list< _Tp >, _Compare)
- template<typename _FIter>
  constexpr pair< _FIter, _FIter > **std::minmax_element** (_FIter, _FIter)
- template<typename _FIter, typename _Compare>
  constexpr pair< _FIter, _FIter > **std::minmax_element** (_FIter, _FIter, _Compare)
- template<typename _IIter1, typename _IIter2>
  constexpr pair< _IIter1, _IIter2 > **std::mismatch** (_IIter1, _IIter1, _IIter2)

- template<typename _IIter1, typename _IIter2, typename _BinaryPredicate>
  constexpr pair< _IIter1, _IIter2 > **std::mismatch** (_IIter1, _IIter1, _IIter2, _BinaryPredicate)
- template<typename _BIter>
  constexpr bool **std::next_permutation** (_BIter, _BIter)
- template<typename _BIter, typename _Compare>
  constexpr bool **std::next_permutation** (_BIter, _BIter, _Compare)
- template<typename _IIter, typename _Predicate>
  constexpr bool **std::none_of** (_IIter, _IIter, _Predicate)
- template<typename _RAIter>
  constexpr void **std::nth_element** (_RAIter, _RAIter, _RAIter)
- template<typename _RAIter, typename _Compare>
  constexpr void **std::nth_element** (_RAIter, _RAIter, _RAIter, _Compare)
- template<typename _RAIter>
  constexpr void **std::partial_sort** (_RAIter, _RAIter, _RAIter)
- template<typename _RAIter, typename _Compare>
  constexpr void **std::partial_sort** (_RAIter, _RAIter, _RAIter, _Compare)
- template<typename _IIter, typename _RAIter>
  constexpr _RAIter **std::partial_sort_copy** (_IIter, _IIter, _RAIter, _RAIter)
- template<typename _IIter, typename _RAIter, typename _Compare>
  constexpr _RAIter **std::partial_sort_copy** (_IIter, _IIter, _RAIter, _RAIter, _Compare)
- template<typename _BIter, typename _Predicate>
  constexpr _BIter **std::partition** (_BIter, _BIter, _Predicate)
- template<typename _IIter, typename _OIter1, typename _OIter2, typename _Predicate>
  constexpr pair< _OIter1, _OIter2 > **std::partition_copy** (_IIter, _IIter, _OIter1, _OIter2, _Predicate)
- template<typename _FIter, typename _Predicate>
  constexpr _FIter **std::partition_point** (_FIter, _FIter, _Predicate)
- template<typename _RAIter>
  constexpr void **std::pop_heap** (_RAIter, _RAIter)
- template<typename _RAIter, typename _Compare>
  constexpr void **std::pop_heap** (_RAIter, _RAIter, _Compare)
- template<typename _BIter>
  constexpr bool **std::prev_permutation** (_BIter, _BIter)
- template<typename _BIter, typename _Compare>
  constexpr bool **std::prev_permutation** (_BIter, _BIter, _Compare)
- template<typename _RAIter>
  constexpr void **std::push_heap** (_RAIter, _RAIter)
- template<typename _RAIter, typename _Compare>
  constexpr void **std::push_heap** (_RAIter, _RAIter, _Compare)
- template<typename _RAIter>
  void **std::random_shuffle** (_RAIter, _RAIter)
- template<typename _RAIter, typename _Generator>
  void **std::random_shuffle** (_RAIter, _RAIter, _Generator &&)
- template<typename _FIter, typename _Tp>
  constexpr _FIter **std::remove** (_FIter, _FIter, const _Tp &)
- template<typename _IIter, typename _OIter, typename _Tp>
  constexpr _OIter **std::remove_copy** (_IIter, _IIter, _OIter, const _Tp &)
- template<typename _IIter, typename _OIter, typename _Predicate>
  constexpr _OIter **std::remove_copy_if** (_IIter, _IIter, _OIter, _Predicate)
- template<typename _FIter, typename _Predicate>
  constexpr _FIter **std::remove_if** (_FIter, _FIter, _Predicate)
- template<typename _FIter, typename _Tp>
  constexpr void **std::replace** (_FIter, _FIter, const _Tp &, const _Tp &)

- template<typename _IIter, typename _OIter, typename _Tp>
  constexpr _OIter **std::replace_copy** (_IIter, _IIter, _OIter, const _Tp &, const _Tp &)
- template<typename _Iter, typename _OIter, typename _Predicate, typename _Tp>
  constexpr _OIter **std::replace_copy_if** (_Iter, _Iter, _OIter, _Predicate, const _Tp &)
- template<typename _FIter, typename _Predicate, typename _Tp>
  constexpr void **std::replace_if** (_FIter, _FIter, _Predicate, const _Tp &)
- template<typename _BIter>
  constexpr void **std::reverse** (_BIter, _BIter)
- template<typename _BIter, typename _OIter>
  constexpr _OIter **std::reverse_copy** (_BIter, _BIter, _OIter)
- template<typename _FIter>
  constexpr _FIter **std::rotate** (_FIter, _FIter, _FIter)
- template<typename _FIter, typename _OIter>
  constexpr _OIter **std::rotate_copy** (_FIter, _FIter, _FIter, _OIter)
- template<typename _FIter1, typename _FIter2>
  constexpr _FIter1 **std::search** (_FIter1, _FIter1, _FIter2, _FIter2)
- template<typename _FIter1, typename _FIter2, typename _BinaryPredicate>
  constexpr _FIter1 **std::search** (_FIter1, _FIter1, _FIter2, _FIter2, _BinaryPredicate)
- template<typename _FIter, typename _Size, typename _Tp>
  constexpr _FIter **std::search_n** (_FIter, _FIter, _Size, const _Tp &)
- template<typename _FIter, typename _Size, typename _Tp, typename _BinaryPredicate>
  constexpr _FIter **std::search_n** (_FIter, _FIter, _Size, const _Tp &, _BinaryPredicate)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  constexpr _OIter **std::set_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare>
  constexpr _OIter **std::set_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  constexpr _OIter **std::set_intersection** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare>
  constexpr _OIter **std::set_intersection** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  constexpr _OIter **std::set_symmetric_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare>
  constexpr _OIter **std::set_symmetric_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  constexpr _OIter **std::set_union** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare>
  constexpr _OIter **std::set_union** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)
- template<typename _RAIter, typename _UGenerator>
  void **std::shuffle** (_RAIter, _RAIter, _UGenerator &&)
- template<typename _RAIter>
  constexpr void **std::sort** (_RAIter, _RAIter)
- template<typename _RAIter, typename _Compare>
  constexpr void **std::sort** (_RAIter, _RAIter, _Compare)
- template<typename _RAIter>
  constexpr void **std::sort_heap** (_RAIter, _RAIter)
- template<typename _RAIter, typename _Compare>
  constexpr void **std::sort_heap** (_RAIter, _RAIter, _Compare)
- template<typename _BIter, typename _Predicate>
  _BIter **std::stable_partition** (_BIter, _BIter, _Predicate)
- template<typename _RAIter>
  void **std::stable_sort** (_RAIter, _RAIter)

- template<typename _RAIter, typename _Compare>
  void **std::stable_sort** (_RAIter, _RAIter, _Compare)
- template<typename _FIter1, typename _FIter2>
  constexpr _FIter2 **std::swap_ranges** (_FIter1, _FIter1, _FIter2)
- template<typename _IIter, typename _OIter, typename _UnaryOperation>
  constexpr _OIter **std::transform** (_IIter, _IIter, _OIter, _UnaryOperation)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _BinaryOperation>
  constexpr _OIter **std::transform** (_IIter1, _IIter1, _IIter2, _OIter, _BinaryOperation)
- template<typename _FIter>
  constexpr _FIter **std::unique** (_FIter, _FIter)
- template<typename _FIter, typename _BinaryPredicate>
  constexpr _FIter **std::unique** (_FIter, _FIter, _BinaryPredicate)
- template<typename _IIter, typename _OIter>
  constexpr _OIter **std::unique_copy** (_IIter, _IIter, _OIter)
- template<typename _IIter, typename _OIter, typename _BinaryPredicate>
  constexpr _OIter **std::unique_copy** (_IIter, _IIter, _OIter, _BinaryPredicate)
- template<typename _FIter, typename _Tp>
  constexpr _FIter **std::upper_bound** (_FIter, _FIter, const _Tp &)
- template<typename _FIter, typename _Tp, typename _Compare>
  constexpr _FIter **std::upper_bound** (_FIter, _FIter, const _Tp &, _Compare)

6.18.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <algorithm>.

6.19 parallel/algorithmfwd.h File Reference

Namespaces

- namespace std
- namespace std::__parallel

Functions

- template<typename _FIter, typename _BiPredicate, typename _IterTag>
  _FIter **std::__parallel::__adjacent_find_switch** (_FIter, _FIter, _BiPredicate, _IterTag)
- template<typename _FIter, typename _IterTag>
  _FIter **std::__parallel::__adjacent_find_switch** (_FIter, _FIter, _IterTag)
- template<typename _RAIter>
  _RAIter **std::__parallel::__adjacent_find_switch** (_RAIter __begin, _RAIter __end, random_access_iterator_tag)
- template<typename _RAIter, typename _BiPredicate>
  _RAIter **std::__parallel::__adjacent_find_switch** (_RAIter, _RAIter, _BiPredicate, random_access_iterator_tag)
- template<typename _IIter, typename _Predicate, typename _IterTag>
  iterator_traits< _IIter >::difference_type **std::__parallel::__count_if_switch** (_IIter, _IIter, _Predicate, _IterTag)
- template<typename _RAIter, typename _Predicate>
  iterator_traits< _RAIter >::difference_type **std::__parallel::__count_if_switch** (_RAIter __begin, _RAIter __↩
  end, _Predicate __pred, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter, typename _Tp, typename _IterTag>
  iterator_traits< _IIter >::difference_type **std::__parallel::__count_switch** (_IIter, _IIter, const _Tp &, _IterTag)
- template<typename _RAIter, typename _Tp>
  iterator_traits< _RAIter >::difference_type **std::__parallel::__count_switch** (_RAIter __begin, _RAIter __end, const _Tp &__value, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)

- template<typename _IIter, typename _FIter, typename _BiPredicate, typename _IterTag1, typename _IterTag2>
  _IIter **std::__parallel::__find_first_of_switch** (_IIter, _IIter, _FIter, _FIter, _BiPredicate, _IterTag1, _IterTag2)
- template<typename _IIter, typename _FIter, typename _IterTag1, typename _IterTag2>
  _IIter **std::__parallel::__find_first_of_switch** (_IIter, _IIter, _FIter, _FIter, _IterTag1, _IterTag2)
- template<typename _RAIter, typename _FIter, typename _BiPredicate, typename _IterTag>
  _RAIter **std::__parallel::__find_first_of_switch** (_RAIter, _RAIter, _FIter, _FIter, _BiPredicate, random_access_iterator_tag, _IterTag)
- template<typename _IIter, typename _Predicate, typename _IterTag>
  _IIter **std::__parallel::__find_if_switch** (_IIter, _IIter, _Predicate, _IterTag)
- template<typename _RAIter, typename _Predicate>
  _RAIter **std::__parallel::__find_if_switch** (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)
- template<typename _IIter, typename _Tp, typename _IterTag>
  _IIter **std::__parallel::__find_switch** (_IIter, _IIter, const _Tp &, _IterTag)
- template<typename _RAIter, typename _Tp>
  _RAIter **std::__parallel::__find_switch** (_RAIter __begin, _RAIter __end, const _Tp &__val, random_access_iterator_tag)
- template<typename _IIter, typename _Function, typename _IterTag>
  _Function **std::__parallel::__for_each_switch** (_IIter, _IIter, _Function, _IterTag)
- template<typename _RAIter, typename _Function>
  _Function **std::__parallel::__for_each_switch** (_RAIter __begin, _RAIter __end, _Function __f, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _OIter, typename _Size, typename _Generator, typename _IterTag>
  _OIter **std::__parallel::__generate_n_switch** (_OIter, _Size, _Generator, _IterTag)
- template<typename _RAIter, typename _Size, typename _Generator>
  _RAIter **std::__parallel::__generate_n_switch** (_RAIter __begin, _Size __n, _Generator __gen, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIter, typename _Generator, typename _IterTag>
  void **std::__parallel::__generate_switch** (_FIter, _FIter, _Generator, _IterTag)
- template<typename _RAIter, typename _Generator>
  void **std::__parallel::__generate_switch** (_RAIter __begin, _RAIter __end, _Generator __gen, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _IterTag1, typename _IterTag2>
  bool **std::__parallel::__lexicographical_compare_switch** (_IIter1, _IIter1, _IIter2, _IIter2, _Predicate, _Iter↩
  Tag1, _IterTag2)
- template<typename _RAIter1, typename _RAIter2, typename _Predicate>
  bool **std::__parallel::__lexicographical_compare_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 _↩
  _begin2, _RAIter2 __end2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _FIter, typename _Compare, typename _IterTag>
  _FIter **std::__parallel::__max_element_switch** (_FIter, _FIter, _Compare, _IterTag)
- template<typename _RAIter, typename _Compare>
  _RAIter **std::__parallel::__max_element_switch** (_RAIter __begin, _RAIter __end, _Compare __comp, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare, typename _IterTag1, typename _IterTag2, typename _IterTag3>
  _OIter **std::__parallel::__merge_switch** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare, _IterTag1, _IterTag2, _IterTag3)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare>
  _OIter **std::__parallel::__merge_switch** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _FIter, typename _Compare, typename _IterTag>
  _FIter **std::__parallel::__min_element_switch** (_FIter, _FIter, _Compare, _IterTag)
- template<typename _RAIter, typename _Compare>
  _RAIter **std::__parallel::__min_element_switch** (_RAIter __begin, _RAIter __end, _Compare __comp, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)

- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _IterTag1, typename _IterTag2>
  pair< _IIter1, _IIter2 > **std::__parallel::__mismatch_switch** (_IIter1, _IIter1, _IIter2, _Predicate, _IterTag1, _IterTag2)
- template<typename _RAIter1, typename _RAIter2, typename _Predicate>
  pair< _RAIter1, _RAIter2 > **std::__parallel::__mismatch_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _FIter, typename _Predicate, typename _IterTag>
  _FIter **std::__parallel::__partition_switch** (_FIter, _FIter, _Predicate, _IterTag)
- template<typename _RAIter, typename _Predicate>
  _RAIter **std::__parallel::__partition_switch** (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)
- template<typename _FIter, typename _Predicate, typename _Tp, typename _IterTag>
  void **std::__parallel::__replace_if_switch** (_FIter, _FIter, _Predicate, const _Tp &, _IterTag)
- template<typename _RAIter, typename _Predicate, typename _Tp>
  void **std::__parallel::__replace_if_switch** (_RAIter __begin, _RAIter __end, _Predicate __pred, const _Tp &__new_value, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIter, typename _Tp, typename _IterTag>
  void **std::__parallel::__replace_switch** (_FIter, _FIter, const _Tp &, const _Tp &, _IterTag)
- template<typename _RAIter, typename _Tp>
  void **std::__parallel::__replace_switch** (_RAIter __begin, _RAIter __end, const _Tp &__old_value, const _Tp &__new_value, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIter, typename _Integer, typename _Tp, typename _BiPredicate, typename _IterTag>
  _FIter **std::__parallel::__search_n_switch** (_FIter, _FIter, _Integer, const _Tp &, _BiPredicate, _IterTag)
- template<typename _RAIter, typename _Integer, typename _Tp, typename _BiPredicate>
  _RAIter **std::__parallel::__search_n_switch** (_RAIter, _RAIter, _Integer, const _Tp &, _BiPredicate, random_access_iterator_tag)
- template<typename _FIter1, typename _FIter2, typename _BiPredicate, typename _IterTag1, typename _IterTag2>
  _FIter1 **std::__parallel::__search_switch** (_FIter1, _FIter1, _FIter2, _FIter2, _BiPredicate, _IterTag1, _IterTag2)
- template<typename _FIter1, typename _FIter2, typename _IterTag1, typename _IterTag2>
  _FIter1 **std::__parallel::__search_switch** (_FIter1, _FIter1, _FIter2, _FIter2, _IterTag1, _IterTag2)
- template<typename _RAIter1, typename _RAIter2>
  _RAIter1 **std::__parallel::__search_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _RAIter1, typename _RAIter2, typename _BiPredicate>
  _RAIter1 **std::__parallel::__search_switch** (_RAIter1, _RAIter1, _RAIter2, _RAIter2, _BiPredicate, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3>
  _OIter **std::__parallel::__set_difference_switch** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)
- template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate>
  _Output_RAIter **std::__parallel::__set_difference_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3>
  _OIter **std::__parallel::__set_intersection_switch** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)
- template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate>
  _Output_RAIter **std::__parallel::__set_intersection_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)

- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3>
  _OIter **std::__parallel::__set_symmetric_difference_switch** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _↩ Predicate, _IterTag1, _IterTag2, _IterTag3)
- template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate>
  _Output_RAIter **std::__parallel::__set_symmetric_difference_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3>
  _OIter **std::__parallel::__set_union_switch** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, _IterTag1, _↩ IterTag2, _IterTag3)
- template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate>
  _Output_RAIter **std::__parallel::__set_union_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 ↩ __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter, typename _OIter, typename _UnaryOperation, typename _IterTag1, typename _IterTag2>
  _OIter **std::__parallel::__transform1_switch** (_IIter, _IIter, _OIter, _UnaryOperation, _IterTag1, _IterTag2)
- template<typename _RAIIter, typename _RAOIter, typename _UnaryOperation>
  _RAOIter **std::__parallel::__transform1_switch** (_RAIIter, _RAIIter, _RAOIter, _UnaryOperation, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism=__gnu_parallel::parallel_balanced)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _BiOperation, typename _Tag1, typename _Tag2, typename _↩ Tag3>
  _OIter **std::__parallel::__transform2_switch** (_IIter1, _IIter1, _IIter2, _OIter, _BiOperation, _Tag1, _Tag2, _↩ Tag3)
- template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BiOperation>
  _RAIter3 **std::__parallel::__transform2_switch** (_RAIter1, _RAIter1, _RAIter2, _RAIter3, _BiOperation, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism=__gnu_parallel::parallel_balanced)
- template<typename _IIter, typename _OIter, typename _Predicate, typename _IterTag1, typename _IterTag2>
  _OIter **std::__parallel::__unique_copy_switch** (_IIter, _IIter, _OIter, _Predicate, _IterTag1, _IterTag2)
- template<typename _RAIter, typename _RandomAccess_OIter, typename _Predicate>
  _RandomAccess_OIter **std::__parallel::__unique_copy_switch** (_RAIter, _RAIter, _RandomAccess_OIter, _↩ Predicate, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _FIter>
  _FIter **std::__parallel::adjacent_find** (_FIter, _FIter)
- template<typename _FIter>
  _FIter **std::__parallel::adjacent_find** (_FIter, _FIter, __gnu_parallel::sequential_tag)
- template<typename _FIter, typename _BiPredicate>
  _FIter **std::__parallel::adjacent_find** (_FIter, _FIter, _BiPredicate)
- template<typename _FIter, typename _BiPredicate>
  _FIter **std::__parallel::adjacent_find** (_FIter, _FIter, _BiPredicate, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _Tp>
  iterator_traits< _IIter >::difference_type **std::__parallel::count** (_IIter __begin, _IIter __end, const _Tp &__↩ value)
- template<typename _IIter, typename _Tp>
  iterator_traits< _IIter >::difference_type **std::__parallel::count** (_IIter __begin, _IIter __end, const _Tp &__↩ value, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter, typename _Tp>
  iterator_traits< _IIter >::difference_type **std::__parallel::count** (_IIter __begin, _IIter __end, const _Tp &__↩ value, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _Predicate>
  iterator_traits< _IIter >::difference_type **std::__parallel::count_if** (_IIter __begin, _IIter __end, _Predicate __↩ pred)

- template<typename _IIter, typename _Predicate>
  iterator_traits< _IIter >::difference_type **std::__parallel::count_if** (_IIter __begin, _IIter __end, _Predicate __pred, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter, typename _Predicate>
  iterator_traits< _IIter >::difference_type **std::__parallel::count_if** (_IIter __begin, _IIter __end, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2>
  constexpr bool **std::__parallel::equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2)
- template<typename _IIter1, typename _IIter2>
  bool **std::__parallel::equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate>
  constexpr bool **std::__parallel::equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred)
- template<typename _IIter1, typename _IIter2, typename _Predicate>
  bool **std::__parallel::equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _Tp>
  _IIter **std::__parallel::find** (_IIter __begin, _IIter __end, const _Tp &__val)
- template<typename _IIter, typename _Tp>
  _IIter **std::__parallel::find** (_IIter __begin, _IIter __end, const _Tp &__val, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _FIter>
  _IIter **std::__parallel::find_first_of** (_IIter, _IIter, _FIter, _FIter)
- template<typename _IIter, typename _FIter>
  _IIter **std::__parallel::find_first_of** (_IIter, _IIter, _FIter, _FIter, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _FIter, typename _BiPredicate>
  _IIter **std::__parallel::find_first_of** (_IIter, _IIter, _FIter, _FIter, _BiPredicate)
- template<typename _IIter, typename _FIter, typename _BiPredicate>
  _IIter **std::__parallel::find_first_of** (_IIter, _IIter, _FIter, _FIter, _BiPredicate, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _Predicate>
  _IIter **std::__parallel::find_if** (_IIter __begin, _IIter __end, _Predicate __pred)
- template<typename _IIter, typename _Predicate>
  _IIter **std::__parallel::find_if** (_IIter __begin, _IIter __end, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _Function>
  _Function **std::__parallel::for_each** (_IIter __begin, _IIter __end, _Function __f, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _Function>
  _Function **std::__parallel::for_each** (_IIter, _IIter, _Function)
- template<typename _Iterator, typename _Function>
  _Function **std::__parallel::for_each** (_Iterator __begin, _Iterator __end, _Function __f, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIter, typename _Generator>
  void **std::__parallel::generate** (_FIter, _FIter, _Generator)
- template<typename _FIter, typename _Generator>
  void **std::__parallel::generate** (_FIter, _FIter, _Generator, __gnu_parallel::_Parallelism)
- template<typename _FIter, typename _Generator>
  void **std::__parallel::generate** (_FIter, _FIter, _Generator, __gnu_parallel::sequential_tag)
- template<typename _OIter, typename _Size, typename _Generator>
  _OIter **std::__parallel::generate_n** (_OIter, _Size, _Generator)
- template<typename _OIter, typename _Size, typename _Generator>
  _OIter **std::__parallel::generate_n** (_OIter, _Size, _Generator, __gnu_parallel::_Parallelism)
- template<typename _OIter, typename _Size, typename _Generator>
  _OIter **std::__parallel::generate_n** (_OIter, _Size, _Generator, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2>
  constexpr bool **std::__parallel::lexicographical_compare** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2)

- template<typename _IIter1, typename _IIter2>
  bool **std::__parallel::lexicographical_compare** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 _↩
  _end2, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate>
  constexpr bool **std::__parallel::lexicographical_compare** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Predicate __pred)
- template<typename _IIter1, typename _IIter2, typename _Predicate>
  bool **std::__parallel::lexicographical_compare** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 _↩
  _end2, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _FIter>
  _FIter **std::__parallel::max_element** (_FIter, _FIter)
- template<typename _FIter>
  _FIter **std::__parallel::max_element** (_FIter, _FIter, __gnu_parallel::_Parallelism)
- template<typename _FIter>
  _FIter **std::__parallel::max_element** (_FIter, _FIter, __gnu_parallel::sequential_tag)
- template<typename _FIter, typename _Compare>
  _FIter **std::__parallel::max_element** (_FIter, _FIter, _Compare)
- template<typename _FIter, typename _Compare>
  _FIter **std::__parallel::max_element** (_FIter, _FIter, _Compare, __gnu_parallel::_Parallelism)
- template<typename _FIter, typename _Compare>
  _FIter **std::__parallel::max_element** (_FIter, _FIter, _Compare, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  _OIter **std::__parallel::merge** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  _OIter **std::__parallel::merge** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare>
  _OIter **std::__parallel::merge** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare>
  _OIter **std::__parallel::merge** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare, __gnu_parallel::sequential_tag)
- template<typename _FIter>
  _FIter **std::__parallel::min_element** (_FIter, _FIter)
- template<typename _FIter>
  _FIter **std::__parallel::min_element** (_FIter, _FIter, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIter>
  _FIter **std::__parallel::min_element** (_FIter, _FIter, __gnu_parallel::sequential_tag)
- template<typename _FIter, typename _Compare>
  _FIter **std::__parallel::min_element** (_FIter, _FIter, _Compare)
- template<typename _FIter, typename _Compare>
  _FIter **std::__parallel::min_element** (_FIter, _FIter, _Compare, __gnu_parallel::_Parallelism)
- template<typename _FIter, typename _Compare>
  _FIter **std::__parallel::min_element** (_FIter, _FIter, _Compare, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2>
  pair< _IIter1, _IIter2 > **std::__parallel::mismatch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2)
- template<typename _IIter1, typename _IIter2>
  pair< _IIter1, _IIter2 > **std::__parallel::mismatch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate>
  pair< _IIter1, _IIter2 > **std::__parallel::mismatch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _↩
  Predicate __pred)
- template<typename _IIter1, typename _IIter2, typename _Predicate>
  pair< _IIter1, _IIter2 > **std::__parallel::mismatch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _↩
  Predicate __pred, __gnu_parallel::sequential_tag)

- template<typename _RAIter>
  void **std::__parallel::nth_element** (_RAIter __begin, _RAIter __nth, _RAIter __end)
- template<typename _RAIter>
  void **std::__parallel::nth_element** (_RAIter __begin, _RAIter __nth, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter, typename _Compare>
  void **std::__parallel::nth_element** (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)
- template<typename _RAIter, typename _Compare>
  void **std::__parallel::nth_element** (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIter>
  void **std::__parallel::partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end)
- template<typename _RAIter>
  void **std::__parallel::partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter, typename _Compare>
  void **std::__parallel::partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)
- template<typename _RAIter, typename _Compare>
  void **std::__parallel::partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _FIter, typename _Predicate>
  _FIter **std::__parallel::partition** (_FIter, _FIter, _Predicate)
- template<typename _FIter, typename _Predicate>
  _FIter **std::__parallel::partition** (_FIter, _FIter, _Predicate, __gnu_parallel::sequential_tag)
- template<typename _RAIter>
  void **std::__parallel::random_shuffle** (_RAIter __begin, _RAIter __end)
- template<typename _RAIter>
  void **std::__parallel::random_shuffle** (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter, typename _RandomNumberGenerator>
  void **std::__parallel::random_shuffle** (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &&__rand)
- template<typename _RAIter, typename _RandomNumberGenerator>
  void **std::__parallel::random_shuffle** (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &__rand, __gnu_parallel::sequential_tag)
- template<typename _FIter, typename _Tp>
  void **std::__parallel::replace** (_FIter, _FIter, const _Tp &, const _Tp &)
- template<typename _FIter, typename _Tp>
  void **std::__parallel::replace** (_FIter, _FIter, const _Tp &, const _Tp &, __gnu_parallel::_Parallelism)
- template<typename _FIter, typename _Tp>
  void **std::__parallel::replace** (_FIter, _FIter, const _Tp &, const _Tp &, __gnu_parallel::sequential_tag)
- template<typename _FIter, typename _Predicate, typename _Tp>
  void **std::__parallel::replace_if** (_FIter, _FIter, _Predicate, const _Tp &)
- template<typename _FIter, typename _Predicate, typename _Tp>
  void **std::__parallel::replace_if** (_FIter, _FIter, _Predicate, const _Tp &, __gnu_parallel::_Parallelism)
- template<typename _FIter, typename _Predicate, typename _Tp>
  void **std::__parallel::replace_if** (_FIter, _FIter, _Predicate, const _Tp &, __gnu_parallel::sequential_tag)
- template<typename _FIter1, typename _FIter2>
  _FIter1 **std::__parallel::search** (_FIter1, _FIter1, _FIter2, _FIter2)
- template<typename _FIter1, typename _FIter2>
  _FIter1 **std::__parallel::search** (_FIter1, _FIter1, _FIter2, _FIter2, __gnu_parallel::sequential_tag)
- template<typename _FIter1, typename _FIter2, typename _BiPredicate>
  constexpr _FIter1 **std::__parallel::search** (_FIter1, _FIter1, _FIter2, _FIter2, _BiPredicate)
- template<typename _FIter1, typename _FIter2, typename _BiPredicate>
  _FIter1 **std::__parallel::search** (_FIter1, _FIter1, _FIter2, _FIter2, _BiPredicate, __gnu_parallel::sequential_tag)
- template<typename _FIter, typename _Integer, typename _Tp>
  _FIter **std::__parallel::search_n** (_FIter, _FIter, _Integer, const _Tp &)

- template<typename _FIter, typename _Integer, typename _Tp>
  _FIter **std::__parallel::search_n** (_FIter, _FIter, _Integer, const _Tp &, __gnu_parallel::sequential_tag)
- template<typename _FIter, typename _Integer, typename _Tp, typename _BiPredicate>
  _FIter **std::__parallel::search_n** (_FIter, _FIter, _Integer, const _Tp &, _BiPredicate)
- template<typename _FIter, typename _Integer, typename _Tp, typename _BiPredicate>
  _FIter **std::__parallel::search_n** (_FIter, _FIter, _Integer, const _Tp &, _BiPredicate, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  _OIter **std::__parallel::set_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  _OIter **std::__parallel::set_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate>
  _OIter **std::__parallel::set_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate>
  _OIter **std::__parallel::set_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  _OIter **std::__parallel::set_intersection** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  _OIter **std::__parallel::set_intersection** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate>
  _OIter **std::__parallel::set_intersection** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate>
  _OIter **std::__parallel::set_intersection** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  _OIter **std::__parallel::set_symmetric_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  _OIter **std::__parallel::set_symmetric_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate>
  _OIter **std::__parallel::set_symmetric_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate>
  _OIter **std::__parallel::set_symmetric_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  _OIter **std::__parallel::set_union** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  _OIter **std::__parallel::set_union** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate>
  _OIter **std::__parallel::set_union** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate>
  _OIter **std::__parallel::set_union** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)
- template<typename _RAIter>
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end)
- template<typename _RAIter>
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter, typename _Compare>
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end, _Compare __comp)
- template<typename _RAIter, typename _Compare>
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIter>
  void **std::__parallel::stable_sort** (_RAIter __begin, _RAIter __end)
- template<typename _RAIter>
  void **std::__parallel::stable_sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter, typename _Compare>
  void **std::__parallel::stable_sort** (_RAIter __begin, _RAIter __end, _Compare __comp)

- template<typename _RAIter, typename _Compare>
  void **std::__parallel::stable_sort** (_RAIter __begin, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OIter, typename _UnaryOperation>
  _OIter **std::__parallel::transform** (_IIter, _IIter, _OIter, _UnaryOperation)
- template<typename _IIter, typename _OIter, typename _UnaryOperation>
  _OIter **std::__parallel::transform** (_IIter, _IIter, _OIter, _UnaryOperation, __gnu_parallel::_Parallelism)
- template<typename _IIter, typename _OIter, typename _UnaryOperation>
  _OIter **std::__parallel::transform** (_IIter, _IIter, _OIter, _UnaryOperation, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _BiOperation>
  _OIter **std::__parallel::transform** (_IIter1, _IIter1, _IIter2, _OIter, _BiOperation)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _BiOperation>
  _OIter **std::__parallel::transform** (_IIter1, _IIter1, _IIter2, _OIter, _BiOperation, __gnu_parallel::_Parallelism)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _BiOperation>
  _OIter **std::__parallel::transform** (_IIter1, _IIter1, _IIter2, _OIter, _BiOperation, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OIter>
  _OIter **std::__parallel::unique_copy** (_IIter, _IIter, _OIter)
- template<typename _IIter, typename _OIter>
  _OIter **std::__parallel::unique_copy** (_IIter, _IIter, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OIter, typename _Predicate>
  _OIter **std::__parallel::unique_copy** (_IIter, _IIter, _OIter, _Predicate)
- template<typename _IIter, typename _OIter, typename _Predicate>
  _OIter **std::__parallel::unique_copy** (_IIter, _IIter, _OIter, _Predicate, __gnu_parallel::sequential_tag)

6.19.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

6.20 align.h File Reference

Namespaces

- namespace std

Functions

- void * std::align (size_t __align, size_t __size, void *&__ptr, size_t &__space) noexcept
- template<size_t _Align, class _Tp>
  constexpr _Tp * std::assume_aligned (_Tp *__ptr) noexcept

6.20.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

6.21 bits/alloc_traits.h File Reference

Classes

- struct std::allocator_traits< _Alloc >
- struct std::allocator_traits< allocator< _Tp > >
- struct std::allocator_traits< allocator< void > >

Namespaces

- namespace std

6.21.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

6.22 ext/alloc_traits.h File Reference

Classes

- struct __gnu_cxx::__alloc_traits< _Alloc, typename >

Namespaces

- namespace __gnu_cxx

6.22.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

6.23 allocated_ptr.h File Reference

Namespaces

- namespace std

6.23.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

6.24 allocator.h File Reference

Classes

- class std::allocator< _Tp >

Namespaces

- namespace std

6.24.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

6.25 atomic_base.h File Reference

Classes

- struct std::atomic_flag

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_ALWAYS_INLINE**
- #define **ATOMIC_FLAG_INIT**
- #define **ATOMIC_VAR_INIT**(_VI)

Typedefs

- typedef unsigned char **std::__atomic_flag_data_type**

Enumerations

- enum class std::memory_order : int {
  relaxed , **consume** , **acquire** , **release** ,
  acq_rel , **seq_cst** }

Functions

- void **std::atomic_signal_fence** (memory_order __m) noexcept
- void **std::atomic_thread_fence** (memory_order __m) noexcept
- template<typename _Tp>
  _Tp std::kill_dependency (_Tp __y) noexcept
- constexpr memory_order **std::operator&** (memory_order __m, __memory_order_modifier __mod) noexcept
- constexpr memory_order **std::operator|** (memory_order __m, __memory_order_modifier __mod) noexcept

Variables

- constexpr memory_order **std::memory_order_acq_rel**
- constexpr memory_order **std::memory_order_acquire**
- constexpr memory_order **std::memory_order_consume**
- constexpr memory_order **std::memory_order_relaxed**
- constexpr memory_order **std::memory_order_release**
- constexpr memory_order **std::memory_order_seq_cst**

6.25.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <atomic>.

6.26 atomic_futex.h File Reference

Namespaces

- namespace std

6.26.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly.

6.27 atomic_lockfree_defines.h File Reference

Macros

- #define ATOMIC_BOOL_LOCK_FREE
- #define **ATOMIC_CHAR16_T_LOCK_FREE**

- #define **ATOMIC_CHAR32_T_LOCK_FREE**
- #define **ATOMIC_CHAR_LOCK_FREE**
- #define **ATOMIC_INT_LOCK_FREE**
- #define **ATOMIC_LLONG_LOCK_FREE**
- #define **ATOMIC_LONG_LOCK_FREE**
- #define **ATOMIC_POINTER_LOCK_FREE**
- #define **ATOMIC_SHORT_LOCK_FREE**
- #define **ATOMIC_WCHAR_T_LOCK_FREE**

6.27.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <atomic>.

6.28 atomic_timed_wait.h File Reference

6.28.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <atomic>.

6.29 atomic_wait.h File Reference

6.29.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <atomic>.

6.30 basic_ios.h File Reference

Classes

- class std::basic_ios< _CharT, _Traits >

Namespaces

- namespace std

Functions

- template<typename _Facet>
  const _Facet & **std::__check_facet** (const _Facet *__f)

6.30.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ios>.

6.31 basic_ios.tcc File Reference

Namespaces

- namespace std

Macros

- #define **_BASIC_IOS_TCC**

6.31.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ios>.

6.32 basic_string.h File Reference

Classes

- class std::basic_string< _CharT, _Traits, _Alloc >
- struct std::hash< basic_string< char, char_traits< char >, _Alloc > >
- struct std::hash< basic_string< char16_t, char_traits< char16_t >, _Alloc > >
- struct std::hash< basic_string< char32_t, char_traits< char32_t >, _Alloc > >
- struct std::hash< basic_string< wchar_t, char_traits< wchar_t >, _Alloc > >

Namespaces

- namespace std
- namespace std::__detail

Functions

- template<typename _Str>
  constexpr _Str **std::__str_concat** (typename _Str::value_type const *__lhs, typename _Str::size_type __lhs_len, typename _Str::value_type const *__rhs, typename _Str::size_type __rhs_len, typename _Str::allocator_type const &__a)
- constexpr void **std::__to_wstring_numeric** (const char *__s, int __len, wchar_t *__wout)
- wstring **std::__to_wstring_numeric** (const string &__s)
- template<typename _InputIterator, typename _CharT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_CharT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  std::basic_string (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> basic_string< _CharT, char_traits< _CharT >, _Allocator >
- template<typename _CharT, typename _Traits, typename _Allocator = allocator<_CharT>, typename = _RequireAllocator<_Allocator>>
  std::basic_string (basic_string_view< _CharT, _Traits >, const _Allocator &=_Allocator()) -> basic_string< _CharT, _Traits, _Allocator >
- template<typename _CharT, typename _Traits, typename _Allocator = allocator<_CharT>, typename = _RequireAllocator<_Allocator>>
  std::basic_string (basic_string_view< _CharT, _Traits >, typename basic_string< _CharT, _Traits, _Allocator >::size_type, typename basic_string< _CharT, _Traits, _Allocator >::size_type, const _Allocator &=_Allocator()) -> basic_string< _CharT, _Traits, _Allocator >
- template<typename _CharT, typename _Traits, typename _Alloc>
  basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &&__is, basic_string< _CharT, _Traits, _Alloc > &__str)
- template<typename _CharT, typename _Traits, typename _Alloc>
  basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &&__is, basic_string< _CharT, _Traits, _Alloc > &__str, _CharT __delim)
- template<typename _CharT, typename _Traits, typename _Alloc>
  basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str)
- template<typename _CharT, typename _Traits, typename _Alloc>
  basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str, _CharT __delim)

- template<> basic_istream< char > & **std::getline** (basic_istream< char > &__in, basic_string< char > &__str, char __delim)
- template<> basic_istream< wchar_t > & **std::getline** (basic_istream< wchar_t > &__in, basic_string< wchar_t > &__str, wchar_t __delim)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > **std::operator+** (_CharT __lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > std::operator+ (_CharT __lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > **std::operator+** (basic_string< _CharT, _Traits, _Alloc > &&↩
  __lhs, _CharT __rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > **std::operator+** (basic_string< _CharT, _Traits, _Alloc > &&↩
  __lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > **std::operator+** (basic_string< _CharT, _Traits, _Alloc > &&↩
  __lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > **std::operator+** (basic_string< _CharT, _Traits, _Alloc > &&↩
  __lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > **std::operator+** (const _CharT *__lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > std::operator+ (const _CharT *__lhs, const basic_string< _↩
  CharT, _Traits, _Alloc > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, _CharT __rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > **std::operator+** (const basic_string< _CharT, _Traits, _Alloc > &__lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, const basic_string< _CharT, _Traits, _Alloc > &__str)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr auto std::operator<=> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs) noexcept -> decltype(__detail::__char_traits_cmp_cat< _Traits >(0))
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr auto std::operator<=> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs) noexcept -> decltype(__detail::__char_traits_cmp_cat< _Traits >(0))
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr bool std::operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr bool std::operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _↩
  CharT, _Traits, _Alloc > &__rhs) noexcept

- template<typename _CharT, typename _Traits, typename _Alloc>
  basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str)
- template<> basic_istream< char > & **std::operator**>> (basic_istream< char > &__is, basic_string< char > &__str)
- double **std::stod** (const string &__str, size_t *__idx=0)
- double **std::stod** (const wstring &__str, size_t *__idx=0)
- float **std::stof** (const string &__str, size_t *__idx=0)
- float **std::stof** (const wstring &__str, size_t *__idx=0)
- int **std::stoi** (const string &__str, size_t *__idx=0, int __base=10)
- int **std::stoi** (const wstring &__str, size_t *__idx=0, int __base=10)
- long **std::stol** (const string &__str, size_t *__idx=0, int __base=10)
- long **std::stol** (const wstring &__str, size_t *__idx=0, int __base=10)
- long double **std::stold** (const string &__str, size_t *__idx=0)
- long double **std::stold** (const wstring &__str, size_t *__idx=0)
- long long **std::stoll** (const string &__str, size_t *__idx=0, int __base=10)
- long long **std::stoll** (const wstring &__str, size_t *__idx=0, int __base=10)
- unsigned long **std::stoul** (const string &__str, size_t *__idx=0, int __base=10)
- unsigned long **std::stoul** (const wstring &__str, size_t *__idx=0, int __base=10)
- unsigned long long **std::stoull** (const string &__str, size_t *__idx=0, int __base=10)
- unsigned long long **std::stoull** (const wstring &__str, size_t *__idx=0, int __base=10)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr void std::swap (basic_string< _CharT, _Traits, _Alloc > &__lhs, basic_string< _CharT, _Traits, _Alloc > &__rhs) noexcept(/*conditional */)
- string **std::to_string** (double __val)
- string **std::to_string** (float __val)
- string **std::to_string** (int __val) noexcept
- string **std::to_string** (long __val) noexcept
- string **std::to_string** (long double __val)
- string **std::to_string** (long long __val)
- string **std::to_string** (unsigned __val) noexcept
- string **std::to_string** (unsigned long __val) noexcept
- string **std::to_string** (unsigned long long __val)
- wstring **std::to_wstring** (double __val)
- wstring **std::to_wstring** (float __val)
- wstring **std::to_wstring** (int __val)
- wstring **std::to_wstring** (long __val)
- wstring **std::to_wstring** (long double __val)
- wstring **std::to_wstring** (long long __val)
- wstring **std::to_wstring** (unsigned __val)
- wstring **std::to_wstring** (unsigned long __val)
- wstring **std::to_wstring** (unsigned long long __val)

6.32.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <string>.

6.33 basic_string.tcc File Reference

Namespaces

- namespace std

Macros

- #define **_BASIC_STRING_TCC**
- #define **_GLIBCXX_STRING_CONSTEXPR**

Functions

- template<typename _CharT, typename _Traits, typename _Alloc>
  basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str, _CharT __delim)
- template<typename _CharT, typename _Traits, typename _Alloc>
  basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str)

6.33.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <string>.

6.34 backward/binders.h File Reference

Classes

- class std::binder1st< _Operation >
- class std::binder2nd< _Operation >

Namespaces

- namespace std

Functions

- template<typename _Operation, typename _Tp>
  binder1st< _Operation > std::bind1st (const _Operation &__fn, const _Tp &__x)
- template<typename _Operation, typename _Tp>
  binder2nd< _Operation > std::bind2nd (const _Operation &__fn, const _Tp &__x)

6.34.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <functional>.

6.35 boost_concept_check.h File Reference

Namespaces

- namespace __gnu_cxx
- namespace __gnu_debug
- namespace std

Macros

- #define **_GLIBCXX_CLASS_REQUIRES**(_type_var, _ns, _concept)
- #define **_GLIBCXX_CLASS_REQUIRES2**(_type_var1, _type_var2, _ns, _concept)
- #define **_GLIBCXX_CLASS_REQUIRES3**(_type_var1, _type_var2, _type_var3, _ns, _concept)
- #define **_GLIBCXX_CLASS_REQUIRES4**(_type_var1, _type_var2, _type_var3, _type_var4, _ns, _concept)

- #define **_GLIBCXX_DEFINE_BINARY_OPERATOR_CONSTRAINT**(_OP, _NAME)
- #define **_GLIBCXX_DEFINE_BINARY_PREDICATE_OP_CONSTRAINT**(_OP, _NAME)
- #define **_IsUnused**

Functions

- template<class _Tp>
  void **__gnu_cxx::__aux_require_boolean_expr** (const _Tp &__t)
- void **__gnu_cxx::__error_type_must_be_a_signed_integer_type** ()
- void **__gnu_cxx::__error_type_must_be_an_integer_type** ()
- void **__gnu_cxx::__error_type_must_be_an_unsigned_integer_type** ()
- template<class _Concept>
  constexpr void **__gnu_cxx::__function_requires** ()

6.35.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iterator>.

6.36 c++0x_warning.h File Reference

6.36.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iosfwd>.

6.37 char_traits.h File Reference

Classes

- struct __gnu_cxx::_Char_types< _CharT >
- struct __gnu_cxx::char_traits< _CharT >
- struct std::char_traits< _CharT >
- struct std::char_traits< wchar_t >

Namespaces

- namespace __gnu_cxx
- namespace std
- namespace std::__detail

Macros

- #define **_GLIBCXX_ALWAYS_INLINE**

Functions

- template<typename _ChTraits>
  constexpr auto **std::__detail::__char_traits_cmp_cat** (int __cmp) noexcept

6.37.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <string>.

6.38 charconv.h File Reference

Namespaces

- namespace std
- namespace std::__detail

Functions

- template<typename _Tp>
  constexpr void **std::__detail::__to_chars_10_impl** (char *__first, unsigned __len, _Tp __val) noexcept
- template<typename _Tp>
  constexpr unsigned **std::__detail::__to_chars_len** (_Tp __value, int __base) noexcept

6.38.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <charconv>.

6.39 chrono.h File Reference

Classes

- struct std::common_type< chrono::duration< _Rep, _Period > >
- struct std::common_type< chrono::duration< _Rep, _Period >, chrono::duration< _Rep, _Period > >
- struct std::common_type< chrono::duration< _Rep1, _Period1 >, chrono::duration< _Rep2, _Period2 > >
- struct std::common_type< chrono::time_point< _Clock, _Duration > >
- struct std::common_type< chrono::time_point< _Clock, _Duration >, chrono::time_point< _Clock, _Duration > >
- struct std::common_type< chrono::time_point< _Clock, _Duration1 >, chrono::time_point< _Clock, _Duration2 > >
- class std::chrono::duration< _Rep, _Period >
- struct std::chrono::duration_values< _Rep >
- struct std::chrono::steady_clock
- struct std::chrono::system_clock
- class std::chrono::time_point< _Clock, _Dur >
- struct std::chrono::treat_as_floating_point< _Rep >

Namespaces

- namespace std
- namespace std::chrono
- namespace std::filesystem
- namespace std::literals
- namespace std::literals::chrono_literals

Typedefs

- using std::chrono::days
- using **std::chrono::file_clock**
- template<typename _Duration>
  using **std::chrono::file_time**
- using std::chrono::high_resolution_clock
- using std::chrono::hours
- using std::chrono::microseconds
- using std::chrono::milliseconds

- using std::chrono::minutes
- using std::chrono::months
- using std::chrono::nanoseconds
- using std::chrono::seconds
- using **std::chrono::sys_days**
- using **std::chrono::sys_seconds**
- template<typename _Duration>
  using **std::chrono::sys_time**
- using std::chrono::weeks
- using std::chrono::years

Functions

- template<typename _Rep, typename _Period>
  constexpr enable_if_t< numeric_limits< _Rep >::is_signed, duration< _Rep, _Period > > std::chrono::abs (duration< _Rep, _Period > __d)
- template<typename _ToDur, typename _Rep, typename _Period>
  constexpr __enable_if_is_duration< _ToDur > std::chrono::__detail::ceil (const duration< _Rep, _Period > &↩ __d)
- template<typename _ToDur, typename _Rep, typename _Period>
  constexpr __enable_if_is_duration< _ToDur > std::chrono::ceil (const duration< _Rep, _Period > &__d)
- template<typename _ToDur, typename _Clock, typename _Dur>
  constexpr enable_if_t< __is_duration_v< _ToDur >, time_point< _Clock, _ToDur > > std::chrono::ceil (const time_point< _Clock, _Dur > &__tp)
- template<typename _ToDur, typename _Rep, typename _Period>
  constexpr __enable_if_is_duration< _ToDur > std::chrono::duration_cast (const duration< _Rep, _Period > &↩ __d)
- template<typename _ToDur, typename _Rep, typename _Period>
  constexpr __enable_if_is_duration< _ToDur > std::chrono::floor (const duration< _Rep, _Period > &__d)
- template<typename _ToDur, typename _Clock, typename _Dur>
  constexpr enable_if_t< __is_duration_v< _ToDur >, time_point< _Clock, _ToDur > > std::chrono::floor (const time_point< _Clock, _Dur > &__tp)
- template<char... _Digits>
  constexpr chrono::hours std::literals::chrono_literals::operator""h ()
- constexpr chrono::duration< long double, ratio< 3600, 1 > > std::literals::chrono_literals::operator""h (long double __hours)
- template<char... _Digits>
  constexpr chrono::minutes std::literals::chrono_literals::operator""min ()
- constexpr chrono::duration< long double, ratio< 60, 1 > > std::literals::chrono_literals::operator""min (long double __mins)
- template<char... _Digits>
  constexpr chrono::milliseconds std::literals::chrono_literals::operator""ms ()
- constexpr chrono::duration< long double, milli > std::literals::chrono_literals::operator""ms (long double __↩ msecs)
- template<char... _Digits>
  constexpr chrono::nanoseconds std::literals::chrono_literals::operator""ns ()
- constexpr chrono::duration< long double, nano > std::literals::chrono_literals::operator""ns (long double __nsecs)
- template<char... _Digits>
  constexpr chrono::seconds std::literals::chrono_literals::operator""s ()
- constexpr chrono::duration< long double > std::literals::chrono_literals::operator""s (long double __secs)
- template<char... _Digits>
  constexpr chrono::microseconds std::literals::chrono_literals::operator""us ()

- constexpr chrono::duration< long double, micro > std::literals::chrono_literals::operator""us (long double __↩ usecs)
- template<typename _ToDur, typename _Rep, typename _Period>
  constexpr enable_if_t< __and_< __is_duration< _ToDur >, __not_< treat_as_floating_point< typename _To↩ Dur::rep > > >::value, _ToDur > std::chrono::round (const duration< _Rep, _Period > &__d)
- template<typename _ToDur, typename _Clock, typename _Dur>
  constexpr enable_if_t< __is_duration_v< _ToDur > &&!treat_as_floating_point_v< typename _ToDur::rep >, time_point< _Clock, _ToDur > > std::chrono::round (const time_point< _Clock, _Dur > &__tp)
- template<typename _ToDur, typename _Clock, typename _Dur>
  constexpr __enable_if_t< __is_duration< _ToDur >::value, time_point< _Clock, _ToDur > > std::chrono::time_point_cast (const time_point< _Clock, _Dur > &__t)

- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type std::chrono::operator- (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)

- template<typename _Rep1, typename _Period, typename _Rep2>
  constexpr duration< __common_rep_t< _Rep1, __disable_if_is_duration< _Rep2 > >, _Period > std::chrono::operator% (const duration< _Rep1, _Period > &__d, const _Rep2 &__s)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type std::chrono::operator% (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Rep2, typename _Period>
  constexpr duration< __common_rep_t< _Rep2, _Rep1 >, _Period > std::chrono::operator* (const _Rep1 &__s, const duration< _Rep2, _Period > &__d)
- template<typename _Rep1, typename _Period, typename _Rep2>
  constexpr duration< __common_rep_t< _Rep1, __disable_if_is_duration< _Rep2 > >, _Period > std::chrono::operator/ (const duration< _Rep1, _Period > &__d, const _Rep2 &__s)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr common_type< _Rep1, _Rep2 >::type std::chrono::operator/ (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)

- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr bool std::chrono::operator< (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr bool std::chrono::operator<= (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  requires three_way_comparable<common_type_t<_Rep1, _Rep2>>
  constexpr auto std::chrono::operator<=> (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr bool std::chrono::operator> (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr bool std::chrono::operator>= (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)

- template<typename _Rep1, typename _Period1, typename _Clock, typename _Dur2>
  constexpr time_point< _Clock, typename common_type< duration< _Rep1, _Period1 >, _Dur2 >::type > std::chrono::operator+ (const duration< _Rep1, _Period1 > &__lhs, const time_point< _Clock, _Dur2 > &↩ __rhs)
- template<typename _Clock, typename _Dur1, typename _Rep2, typename _Period2>
  constexpr time_point< _Clock, typename common_type< _Dur1, duration< _Rep2, _Period2 > >::type > std::chrono::operator- (const time_point< _Clock, _Dur1 > &__lhs, const duration< _Rep2, _Period2 > &_↩ _rhs)
- template<typename _Clock, typename _Dur1, typename _Dur2>
  constexpr common_type< _Dur1, _Dur2 >::type std::chrono::operator- (const time_point< _Clock, _Dur1 > &↩ __lhs, const time_point< _Clock, _Dur2 > &__rhs)

- template<typename _Clock, typename _Dur1, typename _Dur2>
  constexpr bool **std::chrono::operator<** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)
- template<typename _Clock, typename _Dur1, typename _Dur2>
  constexpr bool **std::chrono::operator<=** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _↩ Clock, _Dur2 > &__rhs)
- template<typename _Clock, typename _Dur1, three_way_comparable_with< _Dur1 > _Dur2>
  constexpr auto **std::chrono::operator<=>** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _↩ Clock, _Dur2 > &__rhs)
- template<typename _Clock, typename _Dur1, typename _Dur2>
  constexpr bool **std::chrono::operator>** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)
- template<typename _Clock, typename _Dur1, typename _Dur2>
  constexpr bool **std::chrono::operator>=** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _↩ Clock, _Dur2 > &__rhs)

Variables

- template<typename _Tp, typename = void>
  constexpr bool **std::chrono::is_clock_v**
- template<typename _Tp>
  constexpr bool **std::chrono::is_clock_v< _Tp, void_t< typename _Tp::rep, typename _Tp::period, typename _Tp::duration, typename _Tp::time_point::duration, decltype(_Tp::is_steady), decltype(_Tp↩ ::now())> >**
- template<> constexpr bool **std::chrono::is_clock_v< file_clock >**
- template<> constexpr bool **std::chrono::is_clock_v< steady_clock >**
- template<> constexpr bool **std::chrono::is_clock_v< system_clock >**
- template<typename _Rep>
  constexpr bool **std::chrono::treat_as_floating_point_v**
- template<> constexpr bool **std::chrono::treat_as_floating_point_v< double >**
- template<> constexpr bool **std::chrono::treat_as_floating_point_v< float >**
- template<> constexpr bool **std::chrono::treat_as_floating_point_v< int >**
- template<> constexpr bool **std::chrono::treat_as_floating_point_v< long >**
- template<> constexpr bool **std::chrono::treat_as_floating_point_v< long double >**
- template<> constexpr bool **std::chrono::treat_as_floating_point_v< long long >**

6.39.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <chrono>.

6.39.2 Function Documentation

ceil()

```
template<typename _ToDur, typename _Rep, typename _Period>
__enable_if_is_duration< _ToDur > std::chrono::ceil (
            const duration< _Rep, _Period > & __d)  [nodiscard], [constexpr]
```

Convert a `duration` to type `ToDur` and round up.
If the duration cannot be represented exactly in the result type, returns the closest value that is greater than the argument.

Template Parameters

_ToDur	The result type must be a `duration`.

Parameters

_↩ _d	A duration.

Returns

The value of `__d` converted to type `_ToDur`.

Since

C++17

6.40 chrono_io.h File Reference

Namespaces

- namespace std
- namespace std::chrono

Functions

- void **std::chrono::__detail::from_stream** ()=delete
- template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, day &__d, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Rep, typename _Period, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, duration< _Rep, _Period > &__d, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, file_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, gps_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)

- template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>
basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, local_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>
basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, month &__m, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *_↩
_offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>
basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, month_day &__md, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__↩
offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>
basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, sys_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>
basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, tai_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>
basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, utc_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>
basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, weekday &__wd, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__↩
offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>
basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, year &__y, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>
basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, year_month &__ym, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__↩
offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>
basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, year_month_day &__ymd, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _Duration>
__detail::__local_time_fmt< _Duration > std::chrono::local_time_format (local_time< _Duration > __time, const string *__abbrev=nullptr, const seconds *__offset_sec=nullptr)
- template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const day &__d)
- template<typename _CharT, typename _Traits, typename _Duration>
basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const file_time< _Duration > &__t)
- template<typename _CharT, typename _Traits, typename _Duration>
basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const gps_time< _Duration > &__t)
- template<typename _CharT, typename _Traits, typename _Duration>
basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const hh_mm_ss< _Duration > &__hms)

- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const local_info &__li)
- template<typename _CharT, typename _Traits, typename _Duration>
  requires requires(const sys_time<_Duration>& __st) { __os << __st; }
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const local_time< _Duration > &__lt)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const month &__m)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const month_day &__md)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const month_day_last &__mdl)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const month_weekday &__mwd)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const month_weekday_last &__mwdl)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const sys_days &__dp)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const sys_info &__i)
- template<typename _CharT, typename _Traits, typename _Duration>
  requires (!treat_as_floating_point_v<typename _Duration::rep>) && ratio_less_v<typename _Duration::period, days::period>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const sys_time< _Duration > &__tp)
- template<typename _CharT, typename _Traits, typename _Duration>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const tai_time< _Duration > &__t)
- template<typename _CharT, typename _Traits, typename _Duration>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const utc_time< _Duration > &__t)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const weekday &__wd)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const weekday_indexed &__wdi)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const weekday_last &__wdl)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const year &__y)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const year_month &__ym)

- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const year_month_day &__ymd)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const year_month_day_last &__ymdl)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const year_month_weekday &__ymwd)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const year_month_weekday_last &__ymwdl)
- template<typename _CharT, typename _Traits, typename _Duration, typename _TimeZonePtr>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const zoned_time< _Duration, _TimeZonePtr > &__t)
- template<typename _CharT, typename _Traits, typename _Rep, typename _Period>
  basic_ostream< _CharT, _Traits > & std::chrono::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const duration< _Rep, _Period > &__d)
- template<typename _CharT, __detail::__parsable< _CharT > _Parsable>
  auto **std::chrono::parse** (const _CharT *__fmt, _Parsable &__tp)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits, _Alloc>, __detail::__parsable< _CharT, _Traits, _StrT > _Parsable>
  auto **std::chrono::parse** (const _CharT *__fmt, _Parsable &__tp, basic_string< _CharT, _Traits, _Alloc > &_abbrev)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits, _Alloc>, __detail::__parsable< _CharT, _Traits, _StrT, minutes > _Parsable>
  auto **std::chrono::parse** (const _CharT *__fmt, _Parsable &__tp, basic_string< _CharT, _Traits, _Alloc > &_abbrev, minutes &__offset)
- template<typename _CharT, typename _Traits = char_traits<_CharT>, typename _StrT = basic_string<_CharT, _Traits>, __detail::__parsable< _CharT, _Traits, _StrT, minutes > _Parsable>
  auto **std::chrono::parse** (const _CharT *__fmt, _Parsable &__tp, minutes &__offset)
- template<typename _CharT, typename _Traits, typename _Alloc, __detail::__parsable< _CharT, _Traits > _Parsable>
  auto **std::chrono::parse** (const basic_string< _CharT, _Traits, _Alloc > &__fmt, _Parsable &__tp)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits, _Alloc>, __detail::__parsable< _CharT, _Traits, _StrT > _Parsable>
  auto **std::chrono::parse** (const basic_string< _CharT, _Traits, _Alloc > &__fmt, _Parsable &__tp, basic_string< _CharT, _Traits, _Alloc > &__abbrev)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits, _Alloc>, __detail::__parsable< _CharT, _Traits, _StrT, minutes > _Parsable>
  auto **std::chrono::parse** (const basic_string< _CharT, _Traits, _Alloc > &__fmt, _Parsable &__tp, basic_string< _CharT, _Traits, _Alloc > &__abbrev, minutes &__offset)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits>, __detail::__parsable< _CharT, _Traits, _StrT, minutes > _Parsable>
  auto **std::chrono::parse** (const basic_string< _CharT, _Traits, _Alloc > &__fmt, _Parsable &__tp, minutes &_offset)

6.40.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <chrono>.

6.41 codecvt.h File Reference

Classes

- class std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >
- class std::codecvt< _InternT, _ExternT, _StateT >
- class std::codecvt< char, char, mbstate_t >
- class std::codecvt< char16_t, char, mbstate_t >
- class std::codecvt< char32_t, char, mbstate_t >
- class std::codecvt< wchar_t, char, mbstate_t >
- class std::codecvt_base
- class std::codecvt_byname< _InternT, _ExternT, _StateT >

Namespaces

- namespace std

Functions

- template bool **std::has_facet**< **codecvt**< **char, char, mbstate_t** > > (const locale &)
- template bool **std::has_facet**< **codecvt**< **wchar_t, char, mbstate_t** > > (const locale &)
- template const codecvt< char, char, mbstate_t > & **std::use_facet**< **codecvt**< **char, char, mbstate_t** > > (const locale &)
- template const codecvt< wchar_t, char, mbstate_t > & **std::use_facet**< **codecvt**< **wchar_t, char, mbstate_t** > > (const locale &)

Variables

- template<typename _InternT, typename _ExternT, typename _StateT>
  locale::id **std::codecvt**< **_InternT, _ExternT, _StateT** >**::id**

6.41.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

6.42 concept_check.h File Reference

Macros

- #define **__glibcxx_class_requires**(_a, _b)
- #define **__glibcxx_class_requires2**(_a, _b, _c)
- #define **__glibcxx_class_requires3**(_a, _b, _c, _d)
- #define **__glibcxx_class_requires4**(_a, _b, _c, _d, _e)
- #define **__glibcxx_function_requires**(...)

6.42.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iterator>.

6.43 cow_string.h File Reference

Classes

- class std::basic_string< _CharT, _Traits, _Alloc >

Namespaces

- namespace std

Functions

- template<typename _Operation>
  void **std::basic_string< _CharT, _Traits, _Alloc >::__resize_and_overwrite** (const size_type __n, _Operation __op)

Variables

- template<typename _CharT, typename _Traits, typename _Alloc>
  const basic_string< _CharT, _Traits, _Alloc >::size_type **std::basic_string< _CharT, _Traits, _Alloc >::npos**

6.43.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <string>.
Defines the reference-counted COW string implementation.

6.44 cpp_type_traits.h File Reference

Namespaces

- namespace std

Macros

- #define **__INT_N**(TYPE)

Functions

- template<typename _Iterator>
  constexpr _Iterator **std::__miter_base** (_Iterator __it)

Variables

- template<typename _ValT, typename _Tp>
  constexpr bool **std::__can_use_memchr_for_find**

6.44.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ext/type_traits.h>.

6.45 cpyfunc_impl.h File Reference

Classes

- class std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_MOF_CV_REF**
- #define **_GLIBCXX_MOF_INV_QUALS**
- #define **_GLIBCXX_MOF_REF**

6.45.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <functional>.

6.46 cxxabi_forced.h File Reference

Classes

- class __cxxabiv1::__forced_unwind

6.46.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <cxxabi.h>.

6.47 cxxabi_init_exception.h File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CDTOR_CALLABI**
- #define **_GLIBCXX_HAVE_CDTOR_CALLABI**

Functions

- void * **__cxxabiv1::__cxa_allocate_exception** (size_t) noexcept
- void **__cxxabiv1::__cxa_free_exception** (void *) noexcept
- __cxa_refcounted_exception * **__cxxabiv1::__cxa_init_primary_exception** (void *__object, std::type_info *↩ __tinfo, void(*__dest)(void *)) noexcept

6.47.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly.

6.48 deque.tcc File Reference

Namespaces

- namespace std

Macros

- #define **_DEQUE_TCC**

Functions

- template<bool _IsMove, typename _ITp, typename _IRef, typename _IPtr, typename _OTp>
  ::_Deque_iterator< _OTp, _OTp &, _OTp * > **std::__copy_move_a1** (::_Deque_iterator< _ITp, _IRef, _IPtr > __first, ::_Deque_iterator< _ITp, _IRef, _IPtr > __last, ::_Deque_iterator< _OTp, _OTp &, _OTp * > __result)
- template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI>
  _OI **std::__copy_move_a1** (::_Deque_iterator< _Tp, _Ref, _Ptr > __first, ::_Deque_iterator< _Tp, _Ref, _Ptr > __last, _OI __result)
- template<bool _IsMove, typename _II, typename _Tp>
  __gnu_cxx::__enable_if< __is_random_access_iter< _II >::__value,::_Deque_iterator< _Tp, _Tp &, _Tp * > >::__type **std::__copy_move_a1** (_II __first, _II __last, ::_Deque_iterator< _Tp, _Tp &, _Tp * > __result)
- template<bool _IsMove, typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value,::_Deque_iterator< _CharT, _CharT &, _CharT * > >::__type **std::__copy_move_a2** (istreambuf_iterator< _CharT, char_traits< _CharT > > __first, istreambuf_iterator< _CharT, char_traits< _CharT > > __last, ::_Deque_iterator< _CharT, _CharT &, _CharT * > __result)
- template<bool _IsMove, typename _ITp, typename _IRef, typename _IPtr, typename _OTp>
  ::_Deque_iterator< _OTp, _OTp &, _OTp * > **std::__copy_move_backward_a1** (::_Deque_iterator< _ITp, _IRef, _IPtr > __first, ::_Deque_iterator< _ITp, _IRef, _IPtr > __last, ::_Deque_iterator< _OTp, _OTp &, _OTp * > __result)
- template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI>
  _OI **std::__copy_move_backward_a1** (::_Deque_iterator< _Tp, _Ref, _Ptr > __first, ::_Deque_iterator< _Tp, _Ref, _Ptr > __last, _OI __result)
- template<bool _IsMove, typename _II, typename _Tp>
  __gnu_cxx::__enable_if< __is_random_access_iter< _II >::__value,::_Deque_iterator< _Tp, _Tp &, _Tp * > >::__type **std::__copy_move_backward_a1** (_II __first, _II __last, ::_Deque_iterator< _Tp, _Tp &, _Tp * > __result)
- template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI>
  _OI **std::__copy_move_backward_dit** (::_Deque_iterator< _Tp, _Ref, _Ptr > __first, ::_Deque_iterator< _Tp, _Ref, _Ptr > __last, _OI __result)
- template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI>
  _OI **std::__copy_move_dit** (::_Deque_iterator< _Tp, _Ref, _Ptr > __first, ::_Deque_iterator< _Tp, _Ref, _Ptr > __last, _OI __result)
- template<typename _CharT, typename _Size>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value,::_Deque_iterator< _CharT, _CharT &, _CharT * > >::__type **std::__copy_n_a** (istreambuf_iterator< _CharT, char_traits< _CharT > > __it, _Size __size, ::_Deque_iterator< _CharT, _CharT &, _CharT * > __result, bool __strict)
- template<typename _Tp, typename _Ref, typename _Ptr, typename _II>
  __gnu_cxx::__enable_if< __is_random_access_iter< _II >::__value, bool >::__type **std::__equal_aux1** (::_Deque_iterator< _Tp, _Ref, _Ptr > __first1, ::_Deque_iterator< _Tp, _Ref, _Ptr > __last1, _II __first2)
- template<typename _Tp1, typename _Ref1, typename _Ptr1, typename _Tp2, typename _Ref2, typename _Ptr2>
  bool **std::__equal_aux1** (::_Deque_iterator< _Tp1, _Ref1, _Ptr1 > __first1, ::_Deque_iterator< _Tp1, _Ref1, _Ptr1 > __last1, ::_Deque_iterator< _Tp2, _Ref2, _Ptr2 > __first2)
- template<typename _II, typename _Tp, typename _Ref, typename _Ptr>
  __gnu_cxx::__enable_if< __is_random_access_iter< _II >::__value, bool >::__type **std::__equal_aux1** (_II __first1, _II __last1, ::_Deque_iterator< _Tp, _Ref, _Ptr > __first2)
- template<typename _Tp, typename _Ref, typename _Ptr, typename _II>
  bool **std::__equal_dit** (const ::_Deque_iterator< _Tp, _Ref, _Ptr > &__first1, const ::_Deque_iterator< _Tp, _Ref, _Ptr > &__last1, _II __first2)
- template<typename _Tp, typename _VTp>
  void **std::__fill_a1** (const ::_Deque_iterator< _Tp, _Tp &, _Tp * > &__first, const ::_Deque_iterator< _Tp, _Tp &, _Tp * > &__last, const _VTp &__value)

- template<typename _Tp1, typename _Ref, typename _Ptr, typename _Tp2>
  int **std::__lex_cmp_dit** (::_Deque_iterator< _Tp1, _Ref, _Ptr > __first1, ::_Deque_iterator< _Tp1, _Ref, _Ptr > __last1, const _Tp2 *__first2, const _Tp2 *__last2)
- template<typename _Tp1, typename _Ref1, typename _Ptr1, typename _Tp2, typename _Ref2, typename _Ptr2>
  bool **std::__lexicographical_compare_aux1** (::_Deque_iterator< _Tp1, _Ref1, _Ptr1 > __first1, ::_Deque_iterator< _Tp1, _Ref1, _Ptr1 > __last1, ::_Deque_iterator< _Tp2, _Ref2, _Ptr2 > __first2, ::_Deque_iterator< _Tp2, _Ref2, _Ptr2 > __last2)
- template<typename _Tp1, typename _Ref1, typename _Ptr1, typename _Tp2>
  bool **std::__lexicographical_compare_aux1** (::_Deque_iterator< _Tp1, _Ref1, _Ptr1 > __first1, ::_Deque_iterator< _Tp1, _Ref1, _Ptr1 > __last1, _Tp2 *__first2, _Tp2 *__last2)
- template<typename _Tp1, typename _Tp2, typename _Ref2, typename _Ptr2>
  bool **std::__lexicographical_compare_aux1** (_Tp1 *__first1, _Tp1 *__last1, ::_Deque_iterator< _Tp2, _Ref2, _Ptr2 > __first2, ::_Deque_iterator< _Tp2, _Ref2, _Ptr2 > __last2)

6.48.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <deque>.

6.49 enable_special_members.h File Reference

Namespaces

- namespace std

6.49.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly.

6.50 erase_if.h File Reference

Namespaces

- namespace std
- namespace std::__detail

Functions

- template<typename _Container, typename _UnsafeContainer, typename _Predicate>
  _Container::size_type **std::__detail::__erase_nodes_if** (_Container &__cont, _UnsafeContainer &__ucont, _Predicate __pred)

6.50.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly.

6.51 exception.h File Reference

Classes

- class std::exception

Namespaces

- namespace std

6.51.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly.

6.52 exception_defines.h File Reference

Macros

- #define **__catch**(X)
- #define **__throw_exception_again**
- #define **__try**

6.52.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <exception>.

6.53 exception_ptr.h File Reference

Classes

- class std::__unspecified__::exception_ptr
- class std::exception_ptr

Namespaces

- namespace std

Functions

- exception_ptr std::current_exception () noexcept
- template<typename _Ex>
  exception_ptr std::make_exception_ptr (_Ex __ex) noexcept
- void std::__unspecified__::rethrow_exception (exception_ptr)
- void std::rethrow_exception (exception_ptr)
- void **std::swap** (exception_ptr &__lhs, exception_ptr &__rhs)

6.53.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <exception>.

6.53.2 Function Documentation

rethrow_exception()

```
void std::rethrow_exception (
            exception_ptr )
```

Throw the object pointed to by the exception_ptr.

6.54 formatfwd.h File Reference

6.54.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <format>.

6.55 forward_list.h File Reference

Classes

- struct std::_Fwd_list_base< _Tp, _Alloc >
- struct std::_Fwd_list_const_iterator< _Tp >
- struct std::_Fwd_list_iterator< _Tp >
- struct std::_Fwd_list_node< _Tp >
- struct std::_Fwd_list_node_base
- class std::__fwdlist::_Iterator< _Const, _Ptr >
- struct std::__fwdlist::_Node< _ValPtr >
- struct std::__fwdlist::_Node_base< _VoidPtr >
- class std::forward_list< _Tp, _Alloc >

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_FWDLIST_REMOVE_RETURN_TYPE_TAG**
- #define **_GLIBCXX_USE_ALLOC_PTR_FOR_FWD_LIST**

Functions

- template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  std::forward_list (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> forward_list< _ValT, _Allocator >
- template<typename _Tp, typename _Alloc>
  __detail::__synth3way_t< _Tp > std::operator<=> (const forward_list< _Tp, _Alloc > &__x, const forward_list< _Tp, _Alloc > &__y)
- template<typename _Tp, typename _Alloc>
  bool std::operator== (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)
- template<typename _Tp, typename _Alloc>
  void std::swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly) noexcept(noexcept(__lx.swap(__ly)))

6.55.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <forward_list>.

6.56 forward_list.tcc File Reference

Namespaces

- namespace std

Macros

- #define **_FORWARD_LIST_TCC**
- #define **_GLIBCXX20_ONLY**(__expr)

Functions

- template<typename _Tp, typename _Alloc>
  bool std::operator== (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)

6.56.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <forward_list>.

6.57 bits/fs_dir.h File Reference

Classes

- class std::filesystem::directory_entry
- class std::filesystem::directory_iterator
- class std::filesystem::file_status
- class std::filesystem::recursive_directory_iterator

Namespaces

- namespace std
- namespace std::filesystem

Variables

- template<> constexpr bool **std::ranges::enable_borrowed_range**< **filesystem::directory_iterator** >
- template<> constexpr bool **std::ranges::enable_borrowed_range**< **filesystem::recursive_directory_iterator** >
- template<> constexpr bool **std::ranges::enable_view**< **filesystem::directory_iterator** >
- template<> constexpr bool **std::ranges::enable_view**< **filesystem::recursive_directory_iterator** >

6.57.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <filesystem>.

6.58 experimental/bits/fs_dir.h File Reference

Namespaces

- namespace std
- namespace std::experimental

Functions

- directory_iterator **std::experimental::filesystem::begin** (directory_iterator __iter) noexcept
- recursive_directory_iterator **std::experimental::filesystem::begin** (recursive_directory_iterator __iter) noexcept
- directory_iterator **std::experimental::filesystem::end** (directory_iterator) noexcept
- recursive_directory_iterator **std::experimental::filesystem::end** (recursive_directory_iterator) noexcept
- bool **std::experimental::filesystem::operator!=** (const directory_iterator &__lhs, const directory_iterator &__rhs)
- bool **std::experimental::filesystem::operator!=** (const recursive_directory_iterator &__lhs, const recursive_directory_iterator &__rhs)

- bool **std::experimental::filesystem::operator==** (const directory_iterator &__lhs, const directory_iterator &_↩ _rhs)
- bool **std::experimental::filesystem::operator==** (const recursive_directory_iterator &__lhs, const recursive_↩ directory_iterator &__rhs)

6.58.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <experimental/filesystem>.

6.59 bits/fs_fwd.h File Reference

Classes

- struct std::filesystem::space_info

Namespaces

- namespace std
- namespace std::filesystem

Typedefs

- using std::filesystem::file_time_type

Enumerations

- enum class std::filesystem::copy_options : unsigned short {
  none , **skip_existing** , **overwrite_existing** , **update_existing** ,
  recursive , **copy_symlinks** , **skip_symlinks** , **directories_only** ,
  create_symlinks , **create_hard_links** }
- enum class std::filesystem::directory_options : unsigned char { **none** , **follow_directory_symlink** , **skip_**↩ **permission_denied** }
- enum class std::filesystem::file_type : signed char {
  none , **not_found** , **regular** , **directory** ,
  symlink , **block** , **character** , **fifo** ,
  socket , **unknown** }
- enum class std::filesystem::perm_options : unsigned { **replace** , **add** , **remove** , **nofollow** }
- enum class std::filesystem::perms : unsigned {
  none , **owner_read** , **owner_write** , **owner_exec** ,
  owner_all , **group_read** , **group_write** , **group_exec** ,
  group_all , **others_read** , **others_write** , **others_exec** ,
  others_all , **all** , **set_uid** , **set_gid** ,
  sticky_bit , **mask** , **unknown** }

Functions

- void **std::filesystem::copy** (const path &__from, const path &__to, copy_options __options)
- void **std::filesystem::copy** (const path &__from, const path &__to, copy_options __options, error_code &)
- bool **std::filesystem::copy_file** (const path &__from, const path &__to, copy_options __option)
- bool **std::filesystem::copy_file** (const path &__from, const path &__to, copy_options __option, error_code &)
- path **std::filesystem::current_path** ()
- bool **std::filesystem::exists** (file_status) noexcept
- uintmax_t **std::filesystem::file_size** (const path &)

- uintmax_t **std::filesystem::file_size** (const path &, error_code &) noexcept
- uintmax_t **std::filesystem::hard_link_count** (const path &)
- uintmax_t **std::filesystem::hard_link_count** (const path &, error_code &) noexcept
- bool **std::filesystem::is_other** (file_status) noexcept
- bool **std::filesystem::is_regular_file** (file_status) noexcept
- bool **std::filesystem::is_symlink** (file_status) noexcept
- file_time_type **std::filesystem::last_write_time** (const path &)
- file_time_type **std::filesystem::last_write_time** (const path &, error_code &) noexcept
- copy_options & **std::filesystem::operator&=** (copy_options &__x, copy_options __y) noexcept
- constexpr copy_options **std::filesystem::operator^** (copy_options __x, copy_options __y) noexcept
- copy_options & **std::filesystem::operator^=** (copy_options &__x, copy_options __y) noexcept
- constexpr copy_options **std::filesystem::operator|** (copy_options __x, copy_options __y) noexcept
- copy_options & **std::filesystem::operator|=** (copy_options &__x, copy_options __y) noexcept
- constexpr copy_options **std::filesystem::operator~** (copy_options __x) noexcept
- void **std::filesystem::permissions** (const path &, perms, perm_options, error_code &) noexcept
- path **std::filesystem::proximate** (const path &__p, const path &__base, error_code &__ec)
- path **std::filesystem::relative** (const path &__p, const path &__base, error_code &__ec)
- bool **std::filesystem::remove** (const path &, error_code &) noexcept
- uintmax_t **std::filesystem::remove_all** (const path &)
- uintmax_t **std::filesystem::remove_all** (const path &, error_code &)
- file_status **std::filesystem::status** (const path &)
- file_status **std::filesystem::status** (const path &, error_code &) noexcept
- bool **std::filesystem::status_known** (file_status) noexcept
- file_status **std::filesystem::symlink_status** (const path &)
- file_status **std::filesystem::symlink_status** (const path &, error_code &) noexcept

- perms & **std::filesystem::operator&=** (perms &__x, perms __y) noexcept
- constexpr perms **std::filesystem::operator^** (perms __x, perms __y) noexcept
- perms & **std::filesystem::operator^=** (perms &__x, perms __y) noexcept
- constexpr perms **std::filesystem::operator|** (perms __x, perms __y) noexcept
- perms & **std::filesystem::operator|=** (perms &__x, perms __y) noexcept
- constexpr perms **std::filesystem::operator~** (perms __x) noexcept

- perm_options & **std::filesystem::operator&=** (perm_options &__x, perm_options __y) noexcept
- constexpr perm_options **std::filesystem::operator^** (perm_options __x, perm_options __y) noexcept
- perm_options & **std::filesystem::operator^=** (perm_options &__x, perm_options __y) noexcept
- constexpr perm_options **std::filesystem::operator|** (perm_options __x, perm_options __y) noexcept
- perm_options & **std::filesystem::operator|=** (perm_options &__x, perm_options __y) noexcept
- constexpr perm_options **std::filesystem::operator~** (perm_options __x) noexcept

- directory_options & **std::filesystem::operator&=** (directory_options &__x, directory_options __y) noexcept
- constexpr directory_options **std::filesystem::operator^** (directory_options __x, directory_options __y) noexcept
- directory_options & **std::filesystem::operator^=** (directory_options &__x, directory_options __y) noexcept
- constexpr directory_options **std::filesystem::operator|** (directory_options __x, directory_options __y) noexcept
- directory_options & **std::filesystem::operator|=** (directory_options &__x, directory_options __y) noexcept
- constexpr directory_options **std::filesystem::operator~** (directory_options __x) noexcept

6.59.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <filesystem>.

6.60 experimental/bits/fs_fwd.h File Reference

Classes

- struct std::experimental::filesystem::v1::space_info

Namespaces

- namespace std
- namespace std::experimental

Typedefs

- using std::experimental::filesystem::file_time_type

Enumerations

- enum class std::experimental::filesystem::copy_options : unsigned short {
  none , **skip_existing** , **overwrite_existing** , **update_existing** ,
  recursive , **copy_symlinks** , **skip_symlinks** , **directories_only** ,
  create_symlinks , **create_hard_links** }
- enum class std::experimental::filesystem::directory_options : unsigned char { **none** , **follow_directory_symlink** , **skip_permission_denied** }
- enum class std::experimental::filesystem::file_type : signed char {
  none , **not_found** , **regular** , **directory** ,
  symlink , **block** , **character** , **fifo** ,
  socket , **unknown** }
- enum class std::experimental::filesystem::perms : unsigned {
  none , **owner_read** , **owner_write** , **owner_exec** ,
  owner_all , **group_read** , **group_write** , **group_exec** ,
  group_all , **others_read** , **others_write** , **others_exec** ,
  others_all , **all** , **set_uid** , **set_gid** ,
  sticky_bit , **mask** , **unknown** , **add_perms** ,
  remove_perms , **symlink_nofollow** }

Functions

- void **std::experimental::filesystem::copy** (const path &__from, const path &__to, copy_options __options)
- void **std::experimental::filesystem::copy** (const path &__from, const path &__to, copy_options __options, error_code &) noexcept
- bool **std::experimental::filesystem::copy_file** (const path &__from, const path &__to, copy_options __option)
- bool **std::experimental::filesystem::copy_file** (const path &__from, const path &__to, copy_options __option, error_code &)
- path **std::experimental::filesystem::current_path** ()
- bool **std::experimental::filesystem::is_regular_file** (file_status) noexcept
- bool **std::experimental::filesystem::is_symlink** (file_status) noexcept
- copy_options & **std::experimental::filesystem::operator&=** (copy_options &__x, copy_options __y) noexcept
- constexpr copy_options **std::experimental::filesystem::operator^** (copy_options __x, copy_options __y) noexcept
- copy_options & **std::experimental::filesystem::operator^=** (copy_options &__x, copy_options __y) noexcept

- constexpr copy_options **std::experimental::filesystem::operator|** (copy_options __x, copy_options __y) noexcept
- copy_options & **std::experimental::filesystem::operator|=** (copy_options &__x, copy_options __y) noexcept
- constexpr copy_options **std::experimental::filesystem::operator~** (copy_options __x) noexcept
- file_status **std::experimental::filesystem::status** (const path &)
- file_status **std::experimental::filesystem::status** (const path &, error_code &) noexcept
- bool **std::experimental::filesystem::status_known** (file_status) noexcept
- file_status **std::experimental::filesystem::symlink_status** (const path &)
- file_status **std::experimental::filesystem::symlink_status** (const path &, error_code &) noexcept

- perms & **std::experimental::filesystem::operator&=** (perms &__x, perms __y) noexcept
- constexpr perms **std::experimental::filesystem::operator^** (perms __x, perms __y) noexcept
- perms & **std::experimental::filesystem::operator^=** (perms &__x, perms __y) noexcept
- constexpr perms **std::experimental::filesystem::operator|** (perms __x, perms __y) noexcept
- perms & **std::experimental::filesystem::operator|=** (perms &__x, perms __y) noexcept
- constexpr perms **std::experimental::filesystem::operator~** (perms __x) noexcept

- directory_options & **std::experimental::filesystem::operator&=** (directory_options &__x, directory_options _↩ _y) noexcept
- constexpr directory_options **std::experimental::filesystem::operator^** (directory_options __x, directory_↩ options __y) noexcept
- directory_options & **std::experimental::filesystem::operator^=** (directory_options &__x, directory_options _↩ _y) noexcept
- constexpr directory_options **std::experimental::filesystem::operator|** (directory_options __x, directory_options __y) noexcept
- directory_options & **std::experimental::filesystem::operator|=** (directory_options &__x, directory_options __y) noexcept
- constexpr directory_options **std::experimental::filesystem::operator~** (directory_options __x) noexcept

6.60.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <experimental/filesystem>.

6.61 bits/fs_ops.h File Reference

Namespaces

- namespace std
- namespace std::filesystem

Functions

- path **std::filesystem::absolute** (const path &__p)
- path **std::filesystem::absolute** (const path &__p, error_code &__ec)
- path **std::filesystem::canonical** (const path &__p)
- path **std::filesystem::canonical** (const path &__p, error_code &__ec)
- void **std::filesystem::copy** (const path &__from, const path &__to)
- void **std::filesystem::copy** (const path &__from, const path &__to, copy_options __options)
- void **std::filesystem::copy** (const path &__from, const path &__to, copy_options __options, error_code &)
- void **std::filesystem::copy** (const path &__from, const path &__to, error_code &__ec)

- bool **std::filesystem::copy_file** (const path &__from, const path &__to)
- bool **std::filesystem::copy_file** (const path &__from, const path &__to, copy_options __option)
- bool **std::filesystem::copy_file** (const path &__from, const path &__to, copy_options __option, error_code &)
- bool **std::filesystem::copy_file** (const path &__from, const path &__to, error_code &__ec)
- void **std::filesystem::copy_symlink** (const path &__existing_symlink, const path &__new_symlink)
- void **std::filesystem::copy_symlink** (const path &__existing_symlink, const path &__new_symlink, error_code &__ec) noexcept
- bool **std::filesystem::create_directories** (const path &__p)
- bool **std::filesystem::create_directories** (const path &__p, error_code &__ec)
- bool **std::filesystem::create_directory** (const path &__p)
- bool **std::filesystem::create_directory** (const path &__p, const path &__attributes)
- bool **std::filesystem::create_directory** (const path &__p, const path &__attributes, error_code &__ec) noexcept
- bool **std::filesystem::create_directory** (const path &__p, error_code &__ec) noexcept
- void **std::filesystem::create_directory_symlink** (const path &__to, const path &__new_symlink)
- void **std::filesystem::create_directory_symlink** (const path &__to, const path &__new_symlink, error_code &__ec) noexcept
- void **std::filesystem::create_hard_link** (const path &__to, const path &__new_hard_link)
- void **std::filesystem::create_hard_link** (const path &__to, const path &__new_hard_link, error_code &__ec) noexcept
- void **std::filesystem::create_symlink** (const path &__to, const path &__new_symlink)
- void **std::filesystem::create_symlink** (const path &__to, const path &__new_symlink, error_code &__ec) noexcept
- path **std::filesystem::current_path** ()
- void **std::filesystem::current_path** (const path &__p)
- void **std::filesystem::current_path** (const path &__p, error_code &__ec) noexcept
- path **std::filesystem::current_path** (error_code &__ec)
- bool **std::filesystem::equivalent** (const path &__p1, const path &__p2)
- bool **std::filesystem::equivalent** (const path &__p1, const path &__p2, error_code &__ec) noexcept
- bool **std::filesystem::exists** (const path &__p)
- bool **std::filesystem::exists** (const path &__p, error_code &__ec) noexcept
- bool **std::filesystem::exists** (file_status) noexcept
- uintmax_t **std::filesystem::file_size** (const path &)
- uintmax_t **std::filesystem::file_size** (const path &, error_code &) noexcept
- uintmax_t **std::filesystem::hard_link_count** (const path &)
- uintmax_t **std::filesystem::hard_link_count** (const path &, error_code &) noexcept
- bool **std::filesystem::is_block_file** (const path &__p)
- bool **std::filesystem::is_block_file** (const path &__p, error_code &__ec) noexcept
- bool **std::filesystem::is_block_file** (file_status __s) noexcept
- bool **std::filesystem::is_character_file** (const path &__p)
- bool **std::filesystem::is_character_file** (const path &__p, error_code &__ec) noexcept
- bool **std::filesystem::is_character_file** (file_status __s) noexcept
- bool **std::filesystem::is_directory** (const path &__p)
- bool **std::filesystem::is_directory** (const path &__p, error_code &__ec) noexcept
- bool **std::filesystem::is_directory** (file_status __s) noexcept
- bool **std::filesystem::is_empty** (const path &__p)
- bool **std::filesystem::is_empty** (const path &__p, error_code &__ec)
- bool **std::filesystem::is_fifo** (const path &__p)
- bool **std::filesystem::is_fifo** (const path &__p, error_code &__ec) noexcept
- bool **std::filesystem::is_fifo** (file_status __s) noexcept
- bool **std::filesystem::is_other** (const path &__p)
- bool **std::filesystem::is_other** (const path &__p, error_code &__ec) noexcept

- bool **std::filesystem::is_other** (file_status) noexcept
- bool **std::filesystem::is_regular_file** (const path &__p)
- bool **std::filesystem::is_regular_file** (const path &__p, error_code &__ec) noexcept
- bool **std::filesystem::is_regular_file** (file_status) noexcept
- bool **std::filesystem::is_socket** (const path &__p)
- bool **std::filesystem::is_socket** (const path &__p, error_code &__ec) noexcept
- bool **std::filesystem::is_socket** (file_status __s) noexcept
- bool **std::filesystem::is_symlink** (const path &__p)
- bool **std::filesystem::is_symlink** (const path &__p, error_code &__ec) noexcept
- bool **std::filesystem::is_symlink** (file_status) noexcept
- file_time_type **std::filesystem::last_write_time** (const path &)
- file_time_type **std::filesystem::last_write_time** (const path &, error_code &) noexcept
- void **std::filesystem::last_write_time** (const path &__p, file_time_type __new_time)
- void **std::filesystem::last_write_time** (const path &__p, file_time_type __new_time, error_code &__ec) noexcept
- void **std::filesystem::permissions** (const path &, perms, perm_options, error_code &) noexcept
- void **std::filesystem::permissions** (const path &__p, perms __prms, error_code &__ec) noexcept
- void **std::filesystem::permissions** (const path &__p, perms __prms, perm_options __opts=perm_options↩ ::replace)
- path **std::filesystem::proximate** (const path &__p, const path &__base, error_code &__ec)
- path **std::filesystem::proximate** (const path &__p, const path &__base=current_path())
- path **std::filesystem::proximate** (const path &__p, error_code &__ec)
- path **std::filesystem::read_symlink** (const path &__p)
- path **std::filesystem::read_symlink** (const path &__p, error_code &__ec)
- path **std::filesystem::relative** (const path &__p, const path &__base, error_code &__ec)
- path **std::filesystem::relative** (const path &__p, const path &__base=current_path())
- path **std::filesystem::relative** (const path &__p, error_code &__ec)
- bool **std::filesystem::remove** (const path &, error_code &) noexcept
- bool **std::filesystem::remove** (const path &__p)
- uintmax_t **std::filesystem::remove_all** (const path &)
- uintmax_t **std::filesystem::remove_all** (const path &, error_code &)
- void **std::filesystem::rename** (const path &__from, const path &__to)
- void **std::filesystem::rename** (const path &__from, const path &__to, error_code &__ec) noexcept
- void **std::filesystem::resize_file** (const path &__p, uintmax_t __size)
- void **std::filesystem::resize_file** (const path &__p, uintmax_t __size, error_code &__ec) noexcept
- space_info **std::filesystem::space** (const path &__p)
- space_info **std::filesystem::space** (const path &__p, error_code &__ec) noexcept
- file_status **std::filesystem::status** (const path &)
- file_status **std::filesystem::status** (const path &, error_code &) noexcept
- bool **std::filesystem::status_known** (file_status) noexcept
- file_status **std::filesystem::symlink_status** (const path &)
- file_status **std::filesystem::symlink_status** (const path &, error_code &) noexcept
- path **std::filesystem::temp_directory_path** ()
- path **std::filesystem::temp_directory_path** (error_code &__ec)
- path **std::filesystem::weakly_canonical** (const path &__p)
- path **std::filesystem::weakly_canonical** (const path &__p, error_code &__ec)

6.61.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <filesystem>.

6.62 experimental/bits/fs_ops.h File Reference

Namespaces

- namespace std
- namespace std::experimental

Functions

- path **std::experimental::filesystem::absolute** (const path &__p, const path &__base=current_path())
- path **std::experimental::filesystem::canonical** (const path &__p, const path &__base, error_code &__ec)
- path **std::experimental::filesystem::canonical** (const path &__p, const path &__base=current_path())
- path **std::experimental::filesystem::canonical** (const path &__p, error_code &__ec)
- void **std::experimental::filesystem::copy** (const path &__from, const path &__to)
- void **std::experimental::filesystem::copy** (const path &__from, const path &__to, copy_options __options)
- void **std::experimental::filesystem::copy** (const path &__from, const path &__to, copy_options __options, error_code &__ec) noexcept
- void **std::experimental::filesystem::copy** (const path &__from, const path &__to, error_code &__ec) noexcept
- bool **std::experimental::filesystem::copy_file** (const path &__from, const path &__to)
- bool **std::experimental::filesystem::copy_file** (const path &__from, const path &__to, copy_options __option)
- bool **std::experimental::filesystem::copy_file** (const path &__from, const path &__to, copy_options __option, error_code &__ec)
- bool **std::experimental::filesystem::copy_file** (const path &__from, const path &__to, error_code &__ec)
- void **std::experimental::filesystem::copy_symlink** (const path &__existing_symlink, const path &__new_↩ symlink)
- void **std::experimental::filesystem::copy_symlink** (const path &__existing_symlink, const path &__new_↩ symlink, error_code &__ec) noexcept
- bool **std::experimental::filesystem::create_directories** (const path &__p)
- bool **std::experimental::filesystem::create_directories** (const path &__p, error_code &__ec)
- bool **std::experimental::filesystem::create_directory** (const path &__p)
- bool **std::experimental::filesystem::create_directory** (const path &__p, const path &__attributes)
- bool **std::experimental::filesystem::create_directory** (const path &__p, const path &__attributes, error_code &__ec) noexcept
- bool **std::experimental::filesystem::create_directory** (const path &__p, error_code &__ec) noexcept
- void **std::experimental::filesystem::create_directory_symlink** (const path &__to, const path &__new_symlink)
- void **std::experimental::filesystem::create_directory_symlink** (const path &__to, const path &__new_symlink, error_code &__ec) noexcept
- void **std::experimental::filesystem::create_hard_link** (const path &__to, const path &__new_hard_link)
- void **std::experimental::filesystem::create_hard_link** (const path &__to, const path &__new_hard_link, error_code &__ec) noexcept
- void **std::experimental::filesystem::create_symlink** (const path &__to, const path &__new_symlink)
- void **std::experimental::filesystem::create_symlink** (const path &__to, const path &__new_symlink, error_code &__ec) noexcept
- path **std::experimental::filesystem::current_path** ()
- void **std::experimental::filesystem::current_path** (const path &__p)
- void **std::experimental::filesystem::current_path** (const path &__p, error_code &__ec) noexcept
- path **std::experimental::filesystem::current_path** (error_code &__ec)
- bool **std::experimental::filesystem::equivalent** (const path &__p1, const path &__p2)
- bool **std::experimental::filesystem::equivalent** (const path &__p1, const path &__p2, error_code &__ec) noexcept
- bool **std::experimental::filesystem::exists** (const path &__p)
- bool **std::experimental::filesystem::exists** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::exists** (file_status __s) noexcept

- uintmax_t **std::experimental::filesystem::file_size** (const path &__p)
- uintmax_t **std::experimental::filesystem::file_size** (const path &__p, error_code &__ec) noexcept
- uintmax_t **std::experimental::filesystem::hard_link_count** (const path &__p)
- uintmax_t **std::experimental::filesystem::hard_link_count** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::is_block_file** (const path &__p)
- bool **std::experimental::filesystem::is_block_file** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::is_block_file** (file_status __s) noexcept
- bool **std::experimental::filesystem::is_character_file** (const path &__p)
- bool **std::experimental::filesystem::is_character_file** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::is_character_file** (file_status __s) noexcept
- bool **std::experimental::filesystem::is_directory** (const path &__p)
- bool **std::experimental::filesystem::is_directory** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::is_directory** (file_status __s) noexcept
- bool **std::experimental::filesystem::is_empty** (const path &__p)
- bool **std::experimental::filesystem::is_empty** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::is_fifo** (const path &__p)
- bool **std::experimental::filesystem::is_fifo** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::is_fifo** (file_status __s) noexcept
- bool **std::experimental::filesystem::is_other** (const path &__p)
- bool **std::experimental::filesystem::is_other** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::is_other** (file_status __s) noexcept
- bool **std::experimental::filesystem::is_regular_file** (const path &__p)
- bool **std::experimental::filesystem::is_regular_file** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::is_regular_file** (file_status __s) noexcept
- bool **std::experimental::filesystem::is_socket** (const path &__p)
- bool **std::experimental::filesystem::is_socket** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::is_socket** (file_status __s) noexcept
- bool **std::experimental::filesystem::is_symlink** (const path &__p)
- bool **std::experimental::filesystem::is_symlink** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::is_symlink** (file_status __s) noexcept
- file_time_type **std::experimental::filesystem::last_write_time** (const path &__p)
- file_time_type **std::experimental::filesystem::last_write_time** (const path &__p, error_code &__ec) noexcept
- void **std::experimental::filesystem::last_write_time** (const path &__p, file_time_type __new_time)
- void **std::experimental::filesystem::last_write_time** (const path &__p, file_time_type __new_time, error_code &__ec) noexcept
- void **std::experimental::filesystem::permissions** (const path &__p, perms __prms)
- void **std::experimental::filesystem::permissions** (const path &__p, perms __prms, error_code &__ec) noexcept
- path **std::experimental::filesystem::read_symlink** (const path &__p)
- path **std::experimental::filesystem::read_symlink** (const path &__p, error_code &__ec)
- bool **std::experimental::filesystem::remove** (const path &__p)
- bool **std::experimental::filesystem::remove** (const path &__p, error_code &__ec) noexcept
- uintmax_t **std::experimental::filesystem::remove_all** (const path &__p)
- uintmax_t **std::experimental::filesystem::remove_all** (const path &__p, error_code &__ec)
- void **std::experimental::filesystem::rename** (const path &__from, const path &__to)
- void **std::experimental::filesystem::rename** (const path &__from, const path &__to, error_code &__ec) noexcept
- void **std::experimental::filesystem::resize_file** (const path &__p, uintmax_t __size)
- void **std::experimental::filesystem::resize_file** (const path &__p, uintmax_t __size, error_code &__ec) noexcept
- space_info **std::experimental::filesystem::space** (const path &__p)

- space_info **std::experimental::filesystem::space** (const path &__p, error_code &__ec) noexcept
- file_status **std::experimental::filesystem::status** (const path &__p)
- file_status **std::experimental::filesystem::status** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::status_known** (file_status __s) noexcept
- file_status **std::experimental::filesystem::symlink_status** (const path &__p)
- file_status **std::experimental::filesystem::symlink_status** (const path &__p, error_code &__ec) noexcept
- path **std::experimental::filesystem::system_complete** (const path &__p)
- path **std::experimental::filesystem::system_complete** (const path &__p, error_code &__ec)
- path **std::experimental::filesystem::temp_directory_path** ()
- path **std::experimental::filesystem::temp_directory_path** (error_code &__ec)

6.62.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <experimental/filesystem>.

6.63 bits/fs_path.h File Reference

Classes

- class std::filesystem::filesystem_error
- class std::filesystem::path::iterator
- class std::filesystem::path

Namespaces

- namespace std
- namespace std::filesystem

Functions

- size_t **std::filesystem::hash_value** (const path &__p) noexcept

6.63.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <filesystem>.

6.64 experimental/bits/fs_path.h File Reference

Classes

- class std::experimental::filesystem::v1::basic_string_view< _CharT, _Traits >
- class std::experimental::filesystem::v1::filesystem_error
- class std::experimental::filesystem::v1::path::iterator
- class std::experimental::filesystem::v1::path

Namespaces

- namespace std
- namespace std::experimental

Functions

- bool **std::experimental::filesystem::operator<** (const path &__lhs, const path &__rhs) noexcept
- bool **std::experimental::filesystem::operator==** (const path &__lhs, const path &__rhs) noexcept

6.64.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <experimental/filesystem>.

6.65 fstream.tcc File Reference

Namespaces

- namespace std

Macros

- #define **_FSTREAM_TCC**

6.65.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <fstream>.

6.66 funcref_impl.h File Reference

Classes

- class std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>

Namespaces

- namespace std

6.66.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <functional>.

6.67 functexcept.h File Reference

Namespaces

- namespace std

Functions

- void **std::__throw_bad_alloc** (void)
- void **std::__throw_bad_array_new_length** (void)
- void **std::__throw_bad_cast** (void)
- void **std::__throw_bad_exception** (void)
- void **std::__throw_bad_function_call** ()
- void **std::__throw_bad_typeid** (void)
- void **std::__throw_domain_error** (const char *)
- void **std::__throw_future_error** (int)

- void **std::__throw_invalid_argument** (const char *)
- void **std::__throw_ios_failure** (const char *)
- void **std::__throw_ios_failure** (const char *, int)
- void **std::__throw_length_error** (const char *)
- void **std::__throw_logic_error** (const char *)
- void **std::__throw_out_of_range** (const char *)
- void **std::__throw_out_of_range_fmt** (const char *,...)
- void **std::__throw_overflow_error** (const char *)
- void **std::__throw_range_error** (const char *)
- void **std::__throw_runtime_error** (const char *)
- void **std::__throw_system_error** (int)
- void **std::__throw_underflow_error** (const char *)

6.67.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <exception>.
This header provides support for -fno-exceptions.

6.68 functional_hash.h File Reference

Classes

- struct std::__is_fast_hash< _Hash >
- struct std::hash< _Tp >
- struct std::hash< _Tp * >
- struct std::hash< bool >
- struct std::hash< char >
- struct std::hash< char16_t >
- struct std::hash< char32_t >
- struct std::hash< double >
- struct std::hash< float >
- struct std::hash< int >
- struct std::hash< long >
- struct std::hash< long double >
- struct std::hash< long long >
- struct std::hash< short >
- struct std::hash< signed char >
- struct std::hash< unsigned char >
- struct std::hash< unsigned int >
- struct std::hash< unsigned long >
- struct std::hash< unsigned long long >
- struct std::hash< unsigned short >
- struct std::hash< wchar_t >

Namespaces

- namespace std

Macros

- #define **_Cxx_hashtable_define_trivial_hash**(_Tp)

Variables

- template<typename _Tp, typename = void>
  constexpr bool **std::__is_hash_enabled_for**
- template<typename _Tp>
  constexpr bool **std::__is_hash_enabled_for< _Tp, __void_t< decltype(hash< _Tp >()(declval< _Tp >()))> >**

6.68.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <functional>.

6.69 funcwrap.h File Reference

6.69.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <functional>.

6.70 gslice.h File Reference

Classes

- class std::gslice

Namespaces

- namespace std

6.70.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <valarray>.

6.71 gslice_array.h File Reference

Classes

- class std::gslice_array< _Tp >

Namespaces

- namespace std

6.71.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <valarray>.

6.72 hash_bytes.h File Reference

Namespaces

- namespace std

Functions

- size_t **std::_Fnv_hash_bytes** (const void *__ptr, size_t __len, size_t __seed)
- size_t **std::_Hash_bytes** (const void *__ptr, size_t __len, size_t __seed)

6.72.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <functional>.

6.73 backward/hashtable.h File Reference

Namespaces

- namespace __gnu_cxx

Enumerations

- enum { **_S_num_primes** }

Functions

- unsigned long **__gnu_cxx::__stl_next_prime** (unsigned long __n)
- template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All>
  bool **__gnu_cxx::operator!=** (const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht1, const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht2)
- template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All>
  bool **__gnu_cxx::operator==** (const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht1, const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht2)
- template<class _Val, class _Key, class _HF, class _Extract, class _EqKey, class _All>
  void **__gnu_cxx::swap** (hashtable< _Val, _Key, _HF, _Extract, _EqKey, _All > &__ht1, hashtable< _Val, _Key, _HF, _Extract, _EqKey, _All > &__ht2)

Variables

- template<typename _PrimeType>
  const _PrimeType **__gnu_cxx::_Hashtable_prime_list**< **_PrimeType** >**::__stl_prime_list** [_S_num_primes]

6.73.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

6.74 bits/hashtable.h File Reference

Namespaces

- namespace std

6.74.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <unordered_map> or < unordered_set>.

6.75 hashtable_policy.h File Reference

Namespaces

- namespace std

6.75.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <unordered_map> or <unordered_set>.

6.76 indirect.h File Reference

6.76.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

6.77 indirect_array.h File Reference

Classes

- class std::indirect_array< _Tp >

Namespaces

- namespace std

6.77.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <valarray>.

6.78 invoke.h File Reference

Namespaces

- namespace std

Functions

- template<typename _Tp, typename _Up = typename __inv_unwrap<_Tp>::type>
  constexpr _Up && **std::__invfwd** (typename remove_reference< _Tp >::type &__t) noexcept
- template<typename _Callable, typename... _Args>
  constexpr __invoke_result< _Callable, _Args... >::type std::__invoke (_Callable &&__fn, _Args &&... __args) noexcept(__is_nothrow_invocable< _Callable, _Args... >::value)
- template<typename _Res, typename _MemFun, typename _Tp, typename... _Args>
  constexpr _Res **std::__invoke_impl** (__invoke_memfun_deref, _MemFun &&__f, _Tp &&__t, _Args &&... __args)
- template<typename _Res, typename _MemFun, typename _Tp, typename... _Args>
  constexpr _Res **std::__invoke_impl** (__invoke_memfun_ref, _MemFun &&__f, _Tp &&__t, _Args &&... __args)
- template<typename _Res, typename _MemPtr, typename _Tp>
  constexpr _Res **std::__invoke_impl** (__invoke_memobj_deref, _MemPtr &&__f, _Tp &&__t)
- template<typename _Res, typename _MemPtr, typename _Tp>
  constexpr _Res **std::__invoke_impl** (__invoke_memobj_ref, _MemPtr &&__f, _Tp &&__t)

- template<typename _Res, typename _Fn, typename... _Args>
  constexpr _Res **std::__invoke_impl** (__invoke_other, _Fn &&__f, _Args &&... __args)
- template<typename _Res, typename _Callable, typename... _Args>
  constexpr enable_if_t< is_invocable_r_v< _Res, _Callable, _Args... >, _Res > **std::__invoke_r** (_Callable &&__fn, _Args &&... __args) noexcept(is_nothrow_invocable_r_v< _Res, _Callable, _Args... >)

6.78.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <functional>.

6.79 ios_base.h File Reference

Classes

- class std::ios_base

Namespaces

- namespace std

Macros

- #define **_NOREPLACE_UNUSED**

Enumerations

- enum **_Ios_Fmtflags** {
  _S_boolalpha , **_S_dec** , **_S_fixed** , **_S_hex** ,
  _S_internal , **_S_left** , **_S_oct** , **_S_right** ,
  _S_scientific , **_S_showbase** , **_S_showpoint** , **_S_showpos** ,
  _S_skipws , **_S_unitbuf** , **_S_uppercase** , **_S_adjustfield** ,
  _S_basefield , **_S_floatfield** , **_S_ios_fmtflags_end** , **_S_ios_fmtflags_max** ,
  _S_ios_fmtflags_min }
- enum **_Ios_Iostate** {
  _S_goodbit , **_S_badbit** , **_S_eofbit** , **_S_failbit** ,
  _S_ios_iostate_end , **_S_ios_iostate_max** , **_S_ios_iostate_min** }
- enum **_Ios_Openmode** {
  _S_app , **_S_ate** , **_S_bin** , **_S_in** ,
  _S_out , **_S_trunc** , **_S_noreplace** , **_S_ios_openmode_end** ,
  _S_ios_openmode_max , **_S_ios_openmode_min** }
- enum **_Ios_Seekdir** { **_S_beg** , **_S_cur** , **_S_end** , **_S_ios_seekdir_end** }
- enum class std::io_errc { **stream** }

Functions

- ios_base & std::boolalpha (ios_base &__base)
- ios_base & std::dec (ios_base &__base)
- ios_base & std::defaultfloat (ios_base &__base)
- ios_base & std::fixed (ios_base &__base)
- ios_base & std::hex (ios_base &__base)
- ios_base & std::hexfloat (ios_base &__base)
- ios_base & std::internal (ios_base &__base)
- const error_category & **std::iostream_category** () noexcept

- ios_base & std::left (ios_base &__base)
- error_code **std::make_error_code** (io_errc __e) noexcept
- error_condition **std::make_error_condition** (io_errc __e) noexcept
- ios_base & std::noboolalpha (ios_base &__base)
- ios_base & std::noshowbase (ios_base &__base)
- ios_base & std::noshowpoint (ios_base &__base)
- ios_base & std::noshowpos (ios_base &__base)
- ios_base & std::noskipws (ios_base &__base)
- ios_base & std::nounitbuf (ios_base &__base)
- ios_base & std::nouppercase (ios_base &__base)
- ios_base & std::oct (ios_base &__base)
- constexpr _Ios_Fmtflags **std::operator&** (_Ios_Fmtflags __a, _Ios_Fmtflags __b) noexcept
- constexpr _Ios_Iostate **std::operator&** (_Ios_Iostate __a, _Ios_Iostate __b) noexcept
- constexpr _Ios_Openmode **std::operator&** (_Ios_Openmode __a, _Ios_Openmode __b) noexcept
- constexpr const _Ios_Fmtflags & **std::operator&=** (_Ios_Fmtflags &__a, _Ios_Fmtflags __b) noexcept
- constexpr const _Ios_Iostate & **std::operator&=** (_Ios_Iostate &__a, _Ios_Iostate __b) noexcept
- constexpr const _Ios_Openmode & **std::operator&=** (_Ios_Openmode &__a, _Ios_Openmode __b) noexcept
- constexpr _Ios_Fmtflags **std::operator^** (_Ios_Fmtflags __a, _Ios_Fmtflags __b) noexcept
- constexpr _Ios_Iostate **std::operator^** (_Ios_Iostate __a, _Ios_Iostate __b) noexcept
- constexpr _Ios_Openmode **std::operator^** (_Ios_Openmode __a, _Ios_Openmode __b) noexcept
- constexpr const _Ios_Fmtflags & **std::operator^=** (_Ios_Fmtflags &__a, _Ios_Fmtflags __b) noexcept
- constexpr const _Ios_Iostate & **std::operator^=** (_Ios_Iostate &__a, _Ios_Iostate __b) noexcept
- constexpr const _Ios_Openmode & **std::operator^=** (_Ios_Openmode &__a, _Ios_Openmode __b) noexcept
- constexpr _Ios_Fmtflags **std::operator|** (_Ios_Fmtflags __a, _Ios_Fmtflags __b) noexcept
- constexpr _Ios_Iostate **std::operator|** (_Ios_Iostate __a, _Ios_Iostate __b) noexcept
- constexpr _Ios_Openmode **std::operator|** (_Ios_Openmode __a, _Ios_Openmode __b) noexcept
- constexpr const _Ios_Fmtflags & **std::operator|=** (_Ios_Fmtflags &__a, _Ios_Fmtflags __b) noexcept
- constexpr const _Ios_Iostate & **std::operator|=** (_Ios_Iostate &__a, _Ios_Iostate __b) noexcept
- constexpr const _Ios_Openmode & **std::operator|=** (_Ios_Openmode &__a, _Ios_Openmode __b) noexcept
- constexpr _Ios_Fmtflags **std::operator~** (_Ios_Fmtflags __a) noexcept
- constexpr _Ios_Iostate **std::operator~** (_Ios_Iostate __a) noexcept
- constexpr _Ios_Openmode **std::operator~** (_Ios_Openmode __a) noexcept
- ios_base & std::right (ios_base &__base)
- ios_base & std::scientific (ios_base &__base)
- ios_base & std::showbase (ios_base &__base)
- ios_base & std::showpoint (ios_base &__base)
- ios_base & std::showpos (ios_base &__base)
- ios_base & std::skipws (ios_base &__base)
- ios_base & std::unitbuf (ios_base &__base)
- ios_base & std::uppercase (ios_base &__base)

6.79.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ios>.

6.80 istream.tcc File Reference

Namespaces

- namespace std

Macros

- #define **_ISTREAM_TCC**

Functions

- template<typename _CharT, typename _Traits>
  void **std::__istream_extract** (basic_istream< _CharT, _Traits > &, _CharT *, streamsize)
- template<typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & std::ws (basic_istream< _CharT, _Traits > &__is)
- template<typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__in, _CharT &__c)

6.80.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <istream>.

6.81 iterator_concepts.h File Reference

Classes

- struct std::default_sentinel_t

Namespaces

- namespace std

Variables

- constexpr default_sentinel_t std::default_sentinel

6.81.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iterator>.

6.82 list.tcc File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX20_ONLY**(__expr)
- #define **_LIST_TCC**

6.82.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <list>.

6.83 locale_classes.h File Reference

Classes

- class std::__cxx11::collate< _CharT >
- class std::__cxx11::collate_byname< _CharT >
- class std::locale::facet
- class std::locale::id
- class std::locale

Namespaces

- namespace std

Variables

- template<typename _CharT>
  locale::id **std::collate< _CharT >::id**

6.83.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

6.84 locale_classes.tcc File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_STD_FACET**(...)
- #define **_LOCALE_CLASSES_TCC**

Functions

- template<typename _Facet>
  const _Facet * **std::__try_use_facet** (const locale &__loc)
- template<typename _Facet>
  bool std::has_facet (const locale &__loc)
- template<typename _Facet>
  const _Facet & std::use_facet (const locale &__loc)

6.84.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

6.85 locale_conv.h File Reference

Classes

- class std::wbuffer_convert< _Codecvt, _Elem, _Tr >
- class std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >

Namespaces

- namespace std
- namespace std::__detail

Functions

- template<typename _OutStr, typename _InChar, typename _Codecvt, typename _State, typename _Fn>
  bool **std::__do_str_codecvt** (const _InChar *__first, const _InChar *__last, _OutStr &__outstr, const _Codecvt &__cvt, _State &__state, size_t &__count, _Fn __fn)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _State>
  bool **std::__str_codecvt_in** (const char *__first, const char *__last, basic_string< _CharT, _Traits, _Alloc > &__outstr, const codecvt< _CharT, char, _State > &__cvt)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _State>
  bool **std::__str_codecvt_in** (const char *__first, const char *__last, basic_string< _CharT, _Traits, _Alloc > &__outstr, const codecvt< _CharT, char, _State > &__cvt, _State &__state, size_t &__count)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _State>
  bool **std::__str_codecvt_in_all** (const char *__first, const char *__last, basic_string< _CharT, _Traits, _Alloc > &__outstr, const codecvt< _CharT, char, _State > &__cvt)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _State>
  bool **std::__str_codecvt_out** (const _CharT *__first, const _CharT *__last, basic_string< char, _Traits, _Alloc > &__outstr, const codecvt< _CharT, char, _State > &__cvt)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _State>
  bool **std::__str_codecvt_out** (const _CharT *__first, const _CharT *__last, basic_string< char, _Traits, _Alloc > &__outstr, const codecvt< _CharT, char, _State > &__cvt, _State &__state, size_t &__count)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _State>
  bool **std::__str_codecvt_out_all** (const _CharT *__first, const _CharT *__last, basic_string< char, _Traits, _↩ Alloc > &__outstr, const codecvt< _CharT, char, _State > &__cvt)

6.85.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

6.86 locale_facets.h File Reference

Classes

- class std::__ctype_abstract_base< _CharT >
- class std::ctype< _CharT >
- class std::ctype< char >
- class std::ctype< wchar_t >
- class std::ctype_byname< _CharT >
- class std::ctype_byname< char >
- class std::num_get< _CharT, _InIter >
- class std::num_put< _CharT, _OutIter >
- class std::numpunct< _CharT >
- class std::numpunct_byname< _CharT >

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_NUM_CXX11_FACETS**
- #define **_GLIBCXX_NUM_FACETS**
- #define **_GLIBCXX_NUM_LBDL_ALT128_FACETS**
- #define **_GLIBCXX_NUM_UNICODE_FACETS**

Functions

- template<typename _CharT>
  _CharT * **std::__add_grouping** (_CharT *__s, _CharT __sep, const char *__gbeg, size_t __gsize, const _CharT *__first, const _CharT *__last)
- template<typename _Tp>
  void **std::__convert_to_v** (const char *, _Tp &, ios_base::iostate &, const __c_locale &) throw ()
- template<> void **std::__convert_to_v** (const char *, double &, ios_base::iostate &, const __c_locale &) throw ()
- template<> void **std::__convert_to_v** (const char *, float &, ios_base::iostate &, const __c_locale &) throw ()
- template<> void **std::__convert_to_v** (const char *, long double &, ios_base::iostate &, const __c_locale &) throw ()
- template<typename _CharT, typename _OutIter>
  _OutIter **std::__write** (_OutIter __s, const _CharT *__ws, int __len)
- template<typename _CharT>
  ostreambuf_iterator< _CharT > **std::__write** (ostreambuf_iterator< _CharT > __s, const _CharT *__ws, int __len)
- template<typename _CharT>
  bool std::isalnum (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::isalpha (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::isblank (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::iscntrl (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::isdigit (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::isgraph (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::islower (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::isprint (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::ispunct (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::isspace (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::isupper (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::isxdigit (_CharT __c, const locale &__loc)
- template<typename _CharT>
  _CharT std::tolower (_CharT __c, const locale &__loc)
- template<typename _CharT>
  _CharT std::toupper (_CharT __c, const locale &__loc)

Variables

- template<typename _CharT>
  locale::id **std::ctype< _CharT >::id**
- template<typename _CharT, typename _InIter>
  locale::id **std::num_get< _CharT, _InIter >::id**
- template<typename _CharT, typename _OutIter>
  locale::id **std::num_put< _CharT, _OutIter >::id**
- template<typename _CharT>
  locale::id **std::numpunct< _CharT >::id**

6.86.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

6.87 locale_facets.tcc File Reference

Namespaces

- namespace std

Macros

- #define **_LOCALE_FACETS_TCC**

Functions

- template<typename _CharT>
  _CharT * **std::__add_grouping** (_CharT *__s, _CharT __sep, const char *__gbeg, size_t __gsize, const _CharT *__first, const _CharT *__last)
- template<typename _CharT, typename _ValueT>
  int **std::__int_to_char** (_CharT *__bufend, _ValueT __v, const _CharT *__lit, ios_base::fmtflags __flags, bool __dec)
- bool **std::__verify_grouping** (const char *__grouping, size_t __grouping_size, const string &__grouping_tmp) throw ()

6.87.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

6.88 locale_facets_nonio.h File Reference

Classes

- class std::messages< _CharT >
- struct std::messages_base
- class std::messages_byname< _CharT >
- class std::money_base
- class std::money_get< _CharT, _InIter >
- class std::money_put< _CharT, _OutIter >
- class std::moneypunct< _CharT, _Intl >
- class std::moneypunct_byname< _CharT, _Intl >
- class std::time_base

- class std::time_get< _CharT, _InIter >
- class std::time_get_byname< _CharT, _InIter >
- class std::time_put< _CharT, _OutIter >
- class std::time_put_byname< _CharT, _OutIter >

Namespaces

- namespace std

Variables

- template<typename _CharT>
  locale::id **std::__timepunct< _CharT >::id**
- template<typename _CharT>
  const _CharT * **std::__timepunct_cache< _CharT >::_S_timezones** [14]
- template<> const char * **std::__timepunct_cache< char >::_S_timezones** [14]
- template<> const wchar_t * **std::__timepunct_cache< wchar_t >::_S_timezones** [14]
- template<typename _CharT>
  locale::id **std::messages< _CharT >::id**
- template<typename _CharT, typename _InIter>
  locale::id **std::money_get< _CharT, _InIter >::id**
- template<typename _CharT, typename _OutIter>
  locale::id **std::money_put< _CharT, _OutIter >::id**
- template<typename _CharT, bool _Intl>
  locale::id **std::moneypunct< _CharT, _Intl >::id**
- template<typename _CharT, bool _Intl>
  const bool **std::moneypunct< _CharT, _Intl >::intl**
- template<typename _CharT, bool _Intl>
  const bool **std::moneypunct_byname< _CharT, _Intl >::intl**
- template<typename _CharT, typename _InIter>
  locale::id **std::time_get< _CharT, _InIter >::id**
- template<typename _CharT, typename _OutIter>
  locale::id **std::time_put< _CharT, _OutIter >::id**

6.88.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

6.89 locale_facets_nonio.tcc File Reference

Namespaces

- namespace std

Macros

- #define **_LOCALE_FACETS_NONIO_TCC**

6.89.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

6.90 localefwd.h File Reference

Namespaces

- namespace std

Functions

- template<typename _Facet>
  bool std::has_facet (const locale &__loc)
- template<typename _CharT>
  bool std::isalnum (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::isalpha (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::isblank (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::iscntrl (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::isdigit (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::isgraph (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::islower (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::isprint (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::ispunct (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::isspace (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::isupper (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::isxdigit (_CharT __c, const locale &__loc)
- template<typename _CharT>
  _CharT std::tolower (_CharT __c, const locale &__loc)
- template<typename _CharT>
  _CharT std::toupper (_CharT __c, const locale &__loc)
- template<typename _Facet>
  const _Facet & std::use_facet (const locale &__loc)

6.90.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

6.91 mask_array.h File Reference

Classes

- class std::mask_array< _Tp >

Namespaces

- namespace std

6.91.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <valarray>.

6.92 max_size_type.h File Reference

6.92.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iterator>.

6.93 memory_resource.h File Reference

Classes

- struct std::allocator_traits< pmr::polymorphic_allocator< _Tp > >
- class std::pmr::memory_resource
- class std::pmr::polymorphic_allocator< _Tp >

Namespaces

- namespace std

Functions

- bool **std::pmr::operator==** (const memory_resource &__a, const memory_resource &__b) noexcept
- template<typename _Tp1, typename _Tp2>
  bool **std::pmr::operator==** (const polymorphic_allocator< _Tp1 > &__a, const polymorphic_allocator< _Tp2 > &__b) noexcept

6.93.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory_resource>.

6.94 memoryfwd.h File Reference

Namespaces

- namespace std

6.94.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

6.95 mofunc_impl.h File Reference

Classes

- class std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_MOF_CV_REF**
- #define **_GLIBCXX_MOF_INV_QUALS**
- #define **_GLIBCXX_MOF_REF**

6.95.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <functional>.

6.96 monostate.h File Reference

Namespaces

- namespace std

Functions

- constexpr strong_ordering **std::operator<=>** (monostate, monostate) noexcept
- constexpr bool **std::operator==** (monostate, monostate) noexcept

6.96.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <utility>.

6.97 move.h File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_FORWARD**(_Tp, __val)
- #define **_GLIBCXX_FWDREF**(_Tp)
- #define **_GLIBCXX_MOVE**(__val)

Typedefs

- template<typename _Tp, typename _Up>
  using **std::__like_t**

Functions

- template<typename _Tp>
  constexpr _Tp * std::__addressof (_Tp &__r) noexcept
- template<typename _Tp, typename _Up = _Tp>
  constexpr _Tp **std::__exchange** (_Tp &__obj, _Up &&__new_val)
- template<typename _Tp>
  constexpr _Tp * std::addressof (_Tp &__r) noexcept
- template<typename _Tp>
  const _Tp * **std::addressof** (const _Tp &&)=delete
- template<typename _Tp>
  constexpr _Tp && std::forward (typename std::remove_reference< _Tp >::type &&__t) noexcept

- template<typename _Tp>
  constexpr _Tp && std::forward (typename std::remove_reference< _Tp >::type &__t) noexcept
- template<typename _Tp>
  constexpr std::remove_reference< _Tp >::type && std::move (_Tp &&__t) noexcept
- template<typename _Tp>
  constexpr __conditional_t< __move_if_noexcept_cond< _Tp >::value, const _Tp &, _Tp && > std::move_if_noexcept (_Tp &__x) noexcept
- template<typename _Tp>
  requires (! __is_tuple_like<_Tp>::value) && is_move_constructible_v<_Tp> && is_move_assignable_v<_Tp>
  constexpr void std::swap (_Tp &__a, _Tp &__b) noexcept(/*conditional */) is_nothrow_move_assignable< _Tp > >
- template<typename _Tp, size_t _Nm>
  requires is_swappable_v<_Tp>
  constexpr void std::swap (_Tp(&__a)[_Nm], _Tp(&__b)[_Nm]) noexcept(/*conditional */)

6.97.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <utility>.

6.98 nested_exception.h File Reference

Classes

- class std::nested_exception

Namespaces

- namespace std

Functions

- template<typename _Ex>
  void std::rethrow_if_nested (const _Ex &__ex)
- template<typename _Tp>
  void std::throw_with_nested (_Tp &&__t)

6.98.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <exception>.

6.99 bits/new_allocator.h File Reference

Classes

- class std::__new_allocator< _Tp >

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_OPERATOR_DELETE**
- #define **_GLIBCXX_OPERATOR_NEW**
- #define **_GLIBCXX_SIZED_DEALLOC**(p, n)

6.99.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

6.100 ext/new_allocator.h File Reference

Classes

- class __gnu_cxx::new_allocator< _Tp >

Namespaces

- namespace __gnu_cxx

6.100.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

6.101 node_handle.h File Reference

6.101.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <map,set,unordered_map,unordered_set>.

6.102 ostream.h File Reference

Classes

- class std::basic_ostream< _CharT, _Traits >
- class std::basic_ostream< _CharT, _Traits >::sentry

Namespaces

- namespace std

Typedefs

- template<typename _Os, typename _Tp>
  using **std::__rvalue_stream_insertion_t**

Functions

- template<typename _Ostream, typename _Tp>
  __rvalue_stream_insertion_t< _Ostream, _Tp > std::operator<< (_Ostream &&__os, const _Tp &__x)

- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__out, _CharT __c)

- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__out, char __c)
- template<typename _Traits>
  basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &, char16_t)=delete
- template<typename _Traits>
  basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &, char32_t)=delete
- template<typename _Traits>
  basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &, wchar_t)=delete
- template<typename _Traits>
  basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, char __c)
- template<typename _Traits>
  basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, signed char __c)
- template<typename _Traits>
  basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, unsigned char __c)
- template<typename _Traits>
  basic_ostream< wchar_t, _Traits > & std::operator<< (basic_ostream< wchar_t, _Traits > &, char16_t)=delete
- template<typename _Traits>
  basic_ostream< wchar_t, _Traits > & std::operator<< (basic_ostream< wchar_t, _Traits > &, char32_t)=delete

- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__out, const _CharT *__s)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__out, const char *__s)
- template<typename _Traits>
  basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &, const char16_t *)=delete
- template<typename _Traits>
  basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &, const char32_t *)=delete
- template<typename _Traits>
  basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &, const wchar_t *)=delete
- template<typename _Traits>
  basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, const char *__s)
- template<typename _Traits>
  basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, const signed char *__s)
- template<typename _Traits>
  basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, const unsigned char *__s)
- template<typename _Traits>
  basic_ostream< wchar_t, _Traits > & std::operator<< (basic_ostream< wchar_t, _Traits > &, const char16_t *)=delete
- template<typename _Traits>
  basic_ostream< wchar_t, _Traits > & std::operator<< (basic_ostream< wchar_t, _Traits > &, const char32_t *)=delete

6.102.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ostream>.

6.103 ostream.tcc File Reference

Namespaces

- namespace std

Macros

- #define **_OSTREAM_TCC**

Functions

- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__out, const char *__s)

6.103.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ostream>.

6.104 ostream_insert.h File Reference

Namespaces

- namespace std

6.104.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ostream>.

6.105 out_ptr.h File Reference

6.105.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

6.106 parse_numbers.h File Reference

Namespaces

- namespace std

Typedefs

- template<unsigned long long _Val>
  using **std::__parse_int::__ull_constant**
- template<char... _Digs>
  using **std::__select_int::_Select_int**

6.106.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <chrono>.

6.107 postypes.h File Reference

Classes

- class std::fpos< _StateT >

Namespaces

- namespace std

Typedefs

- typedef long long std::streamoff
- typedef fpos< mbstate_t > std::streampos
- typedef ptrdiff_t std::streamsize
- typedef fpos< mbstate_t > std::u16streampos
- typedef fpos< mbstate_t > std::u32streampos
- typedef fpos< mbstate_t > std::wstreampos

Functions

- template<typename _StateT>
  bool **std::operator!=** (const fpos< _StateT > &__lhs, const fpos< _StateT > &__rhs)
- template<typename _StateT>
  bool std::operator== (const fpos< _StateT > &__lhs, const fpos< _StateT > &__rhs)

6.107.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iosfwd>.

6.108 predefined_ops.h File Reference

Namespaces

- namespace __gnu_cxx

Typedefs

- template<typename _Fn>
  using **__gnu_cxx::__ops::__by_ref_or_value_fn**
- template<typename _Func, typename _Value>
  using **__gnu_cxx::__ops::_Comp_with_val_1st**
- template<typename _Func, typename _Value>
  using **__gnu_cxx::__ops::_Comp_with_val_2nd**
- typedef std::equal_to< void > **__gnu_cxx::__ops::equal_to**
- typedef std::less< void > **__gnu_cxx::__ops::less**

Functions

- template<typename _Value>
  constexpr _Comp_with_val_2nd< equal_to, _Value > **__gnu_cxx::__ops::__equal_to** (const _Value &__val)
- template<typename _Func, typename _Value>
  constexpr _Comp_with_val_1st< _Func, _Value > **__gnu_cxx::__ops::bind1st** (_Func &__f, const _Value &↩
  __val)

- template<typename _Func, typename _Value>
  constexpr _Comp_with_val_2nd< _Func, _Value > **__gnu_cxx::__ops::bind2nd** (_Func &__f, const _Value &__val)
- template<typename _Func>
  constexpr _Unary_negate< _Func > **__gnu_cxx::__ops::not1** (_Func &__f)

6.108.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly. Instead, include <algorithm>.

6.109 ptr_traits.h File Reference

Classes

- struct std::pointer_traits< _Ptr >
- struct std::pointer_traits< _Tp * >

Namespaces

- namespace __gnu_debug
- namespace std

Typedefs

- template<typename _Ptr, typename _Tp>
  using std::__ptr_rebind

Functions

- template<typename _Tp>
  constexpr _Tp * std::to_address (_Tp *__ptr) noexcept
- template<typename _Ptr>
  constexpr auto std::to_address (const _Ptr &__ptr) noexcept

6.109.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

6.110 quoted_string.h File Reference

Classes

- struct std::__detail::_Quoted_string< _String, _CharT >

Namespaces

- namespace std
- namespace std::__detail

Functions

- template<typename _CharT, typename _Traits, typename _String>
  std::basic_ostream< _CharT, _Traits > & std::__detail::operator<< (std::basic_ostream< _CharT, _Traits > &↩
  __os, const _Quoted_string< _String, _CharT > &__str)
- template<typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::__detail::operator<< (std::basic_ostream< _CharT, _Traits > &↩
  __os, const _Quoted_string< const _CharT *, _CharT > &__str)
- template<typename _CharT, typename _Traits, typename _Alloc>
  std::basic_istream< _CharT, _Traits > & std::__detail::operator>> (std::basic_istream< _CharT, _Traits > &_↩
  _is, const _Quoted_string< basic_string< _CharT, _Traits, _Alloc > &, _CharT > &__str)

6.110.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iomanip>.

6.111 random.h File Reference

Classes

- class std::bernoulli_distribution
- class std::binomial_distribution< _IntType >
- class std::cauchy_distribution< _RealType >
- class std::chi_squared_distribution< _RealType >
- class std::discard_block_engine< _RandomNumberEngine, __p, __r >
- class std::discrete_distribution< _IntType >
- class std::exponential_distribution< _RealType >
- class std::extreme_value_distribution< _RealType >
- class std::fisher_f_distribution< _RealType >
- class std::gamma_distribution< _RealType >
- class std::geometric_distribution< _IntType >
- class std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >
- class std::linear_congruential_engine< _UIntType, __a, __c, __m >
- class std::lognormal_distribution< _RealType >
- class std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >
- class std::negative_binomial_distribution< _IntType >
- class std::normal_distribution< _RealType >
- struct std::uniform_real_distribution< _RealType >::param_type
- struct std::normal_distribution< _RealType >::param_type
- struct std::lognormal_distribution< _RealType >::param_type
- struct std::gamma_distribution< _RealType >::param_type
- struct std::chi_squared_distribution< _RealType >::param_type
- struct std::cauchy_distribution< _RealType >::param_type
- struct std::fisher_f_distribution< _RealType >::param_type
- struct std::student_t_distribution< _RealType >::param_type
- struct std::bernoulli_distribution::param_type
- struct std::binomial_distribution< _IntType >::param_type
- struct std::geometric_distribution< _IntType >::param_type
- struct std::negative_binomial_distribution< _IntType >::param_type
- struct std::poisson_distribution< _IntType >::param_type
- struct std::exponential_distribution< _RealType >::param_type
- struct std::weibull_distribution< _RealType >::param_type

- struct std::extreme_value_distribution< _RealType >::param_type
- struct std::discrete_distribution< _IntType >::param_type
- struct std::piecewise_constant_distribution< _RealType >::param_type
- struct std::piecewise_linear_distribution< _RealType >::param_type
- class std::piecewise_constant_distribution< _RealType >
- class std::piecewise_linear_distribution< _RealType >
- class std::poisson_distribution< _IntType >
- class std::random_device
- class std::seed_seq
- class std::shuffle_order_engine< _RandomNumberEngine, __k >
- class std::student_t_distribution< _RealType >
- class std::subtract_with_carry_engine< _UIntType, __w, __s, __r >
- class std::uniform_real_distribution< _RealType >
- class std::weibull_distribution< _RealType >

Namespaces

- namespace std

Typedefs

- typedef minstd_rand0 **std::default_random_engine**
- typedef shuffle_order_engine< minstd_rand0, 256 > **std::knuth_b**
- typedef linear_congruential_engine< uint_fast32_t, 48271UL, 0UL, 2147483647UL > std::minstd_rand
- typedef linear_congruential_engine< uint_fast32_t, 16807UL, 0UL, 2147483647UL > std::minstd_rand0
- typedef mersenne_twister_engine< uint_fast32_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL > std::mt19937
- typedef mersenne_twister_engine< uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67fffeda60000ULL, 37, 0xfff7eee000000000ULL, 43, 6364136223846793005ULL > std::mt19937_64
- typedef discard_block_engine< ranlux24_base, 223, 23 > **std::ranlux24**
- typedef subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 > **std::ranlux24_base**
- typedef discard_block_engine< ranlux48_base, 389, 11 > **std::ranlux48**
- typedef subtract_with_carry_engine< uint_fast64_t, 48, 5, 12 > **std::ranlux48_base**

Functions

- template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator>
  _RealType std::generate_canonical (_UniformRandomNumberGenerator &__g)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_int_distribution< _IntType > &)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_real_distribution< _RealType > &)
- template<typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::bernoulli_distribution &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::cauchy_distribution< _RealType > &__x)

- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::exponential_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::extreme_value_distribution< _RealType > &__x)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::geometric_distribution< _IntType > &__x)
- template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::weibull_distribution< _RealType > &__x)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_int_distribution< _IntType > &)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_real_distribution< _RealType > &)
- template<typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::bernoulli_distribution &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::cauchy_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::exponential_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::extreme_value_distribution< _RealType > &__x)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::geometric_distribution< _IntType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::weibull_distribution< _RealType > &__x)

6.111.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <random>.

6.112 bits/random.tcc File Reference

Namespaces

- namespace std
- namespace std::__detail

Macros

- #define **_RANDOM_TCC**

Functions

- template<typename _ValT, typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & **std::__detail::__extract_params** (basic_istream< _CharT, _Traits > &__is, vector< _ValT > &__vals, size_t __n)
- template<typename _Tp>
  constexpr bool **std::__detail::__p1_representable_as_double** (_Tp __x) noexcept
- template<typename _Tp>
  constexpr bool **std::__detail::__representable_as_double** (_Tp __x) noexcept
- template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator>
  _RealType std::generate_canonical (_UniformRandomNumberGenerator &__g)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_int_distribution< _IntType > &)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_real_distribution< _RealType > &)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **std::operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const binomial_distribution< _IntType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **std::operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const chi_squared_distribution< _RealType > &__x)
- template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **std::operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const discard_block_engine< _RandomNumberEngine, __p, __r > &__x)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **std::operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const discrete_distribution< _IntType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **std::operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const fisher_f_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **std::operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const gamma_distribution< _RealType > &__x)
- template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **std::operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const linear_congruential_engine< _UIntType, __a, __c, __m > &__lcr)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **std::operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const lognormal_distribution< _RealType > &__x)
- template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _↩
  UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **std::operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__x)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **std::operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const negative_binomial_distribution< _IntType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **std::operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const normal_distribution< _RealType > &__x)

- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **std::operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const piecewise_constant_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **std::operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const piecewise_linear_distribution< _RealType > &__x)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **std::operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const poisson_distribution< _IntType > &__x)
- template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **std::operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const shuffle_order_engine< _RandomNumberEngine, __k > &__x)
- template<typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::bernoulli_distribution &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::cauchy_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::exponential_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::extreme_value_distribution< _RealType > &__x)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::geometric_distribution< _IntType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::weibull_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **std::operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const student_t_distribution< _RealType > &__x)
- template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **std::operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const subtract_with_carry_engine< _UIntType, __w, __s, __r > &__x)
- template<typename _RealType>
  bool **std::operator==** (const std::normal_distribution< _RealType > &__d1, const std::normal_distribution< _RealType > &__d2)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_int_distribution< _IntType > &)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_real_distribution< _RealType > &)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **std::operator**>> (std::basic_istream< _CharT, _Traits > &__is, binomial_distribution< _IntType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **std::operator**>> (std::basic_istream< _CharT, _Traits > &__is, chi_squared_distribution< _RealType > &__x)
- template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **std::operator**>> (std::basic_istream< _CharT, _Traits > &__is, discard_block_engine< _RandomNumberEngine, __p, __r > &__x)

- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **std::operator>>** (std::basic_istream< _CharT, _Traits > &__↩
  is, discrete_distribution< _IntType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **std::operator>>** (std::basic_istream< _CharT, _Traits > &__↩
  is, fisher_f_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **std::operator>>** (std::basic_istream< _CharT, _Traits > &__↩
  is, gamma_distribution< _RealType > &__x)
- template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **std::operator>>** (std::basic_istream< _CharT, _Traits > &__↩
  is, linear_congruential_engine< _UIntType, __a, __c, __m > &__lcr)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **std::operator>>** (std::basic_istream< _CharT, _Traits > &__↩
  is, lognormal_distribution< _RealType > &__x)
- template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _↩
  UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **std::operator>>** (std::basic_istream< _CharT, _Traits > &__↩
  is, mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >
  &__x)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **std::operator>>** (std::basic_istream< _CharT, _Traits > &__↩
  is, negative_binomial_distribution< _IntType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **std::operator>>** (std::basic_istream< _CharT, _Traits > &__↩
  is, normal_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **std::operator>>** (std::basic_istream< _CharT, _Traits > &__↩
  is, piecewise_constant_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **std::operator>>** (std::basic_istream< _CharT, _Traits > &__↩
  is, piecewise_linear_distribution< _RealType > &__x)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **std::operator>>** (std::basic_istream< _CharT, _Traits > &__↩
  is, poisson_distribution< _IntType > &__x)
- template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **std::operator>>** (std::basic_istream< _CharT, _Traits > &__↩
  is, shuffle_order_engine< _RandomNumberEngine, __k > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,
  std::cauchy_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,
  std::exponential_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,
  std::extreme_value_distribution< _RealType > &__x)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,
  std::geometric_distribution< _IntType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,
  std::weibull_distribution< _RealType > &__x)

- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **std::operator>>** (std::basic_istream< _CharT, _Traits > &__↩
  is, student_t_distribution< _RealType > &__x)
- template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **std::operator>>** (std::basic_istream< _CharT, _Traits > &__↩
  is, subtract_with_carry_engine< _UIntType, __w, __s, __r > &__x)

6.112.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <random>.

6.113 ext/random.tcc File Reference

Namespaces

- namespace __gnu_cxx

Macros

- #define **_EXT_RANDOM_TCC**

Functions

- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &↩
  __os, const __gnu_cxx::beta_distribution< _RealType > &__x)
- template<typename _UIntType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &↩
  __os, const __gnu_cxx::hypergeometric_distribution< _UIntType > &__x)
- template<size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &↩
  __os, const __gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__x)
- template<typename _UIntType, size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t __msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &↩
  __os, const __gnu_cxx::triangular_distribution< _RealType > &__x)
- template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &↩
  __os, const __gnu_cxx::uniform_inside_sphere_distribution< _Dimen, _RealType > &__x)
- template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &↩
  __os, const __gnu_cxx::uniform_on_sphere_distribution< _Dimen, _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &↩
  __os, const __gnu_cxx::von_mises_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &↩
  __os, const arcsine_distribution< _RealType > &__x)

- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &↩
  __os, const hoyt_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &↩
  __os, const k_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &↩
  __os, const logistic_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &↩
  __os, const nakagami_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &↩
  __os, const pareto_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &↩
  __os, const rice_distribution< _RealType > &__x)
- template<size_t _Dimen, typename _RealType>
  bool **__gnu_cxx::operator==** (const __gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__d1, const
  __gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__d2)
- template<typename _UIntType, size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t
  __msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4>
  bool **__gnu_cxx::operator==** (const __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __↩
  pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4
  > &__lhs, const __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __↩
  sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__rhs)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **__gnu_cxx::operator>>** (std::basic_istream< _CharT, _Traits > &↩
  __is, __gnu_cxx::beta_distribution< _RealType > &__x)
- template<typename _UIntType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **__gnu_cxx::operator>>** (std::basic_istream< _CharT, _Traits > &↩
  __is, __gnu_cxx::hypergeometric_distribution< _UIntType > &__x)
- template<size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **__gnu_cxx::operator>>** (std::basic_istream< _CharT, _Traits > &↩
  __is, __gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__x)
- template<typename _UIntType, size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t
  __msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4, typename _CharT,
  typename _Traits>
  std::basic_istream< _CharT, _Traits > & **__gnu_cxx::operator>>** (std::basic_istream< _CharT, _Traits > &↩
  __is, __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2,
  __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **__gnu_cxx::operator>>** (std::basic_istream< _CharT, _Traits > &↩
  __is, __gnu_cxx::triangular_distribution< _RealType > &__x)
- template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **__gnu_cxx::operator>>** (std::basic_istream< _CharT, _Traits > &↩
  __is, __gnu_cxx::uniform_inside_sphere_distribution< _Dimen, _RealType > &__x)
- template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **__gnu_cxx::operator>>** (std::basic_istream< _CharT, _Traits > &↩
  __is, __gnu_cxx::uniform_on_sphere_distribution< _Dimen, _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **__gnu_cxx::operator>>** (std::basic_istream< _CharT, _Traits > &↩
  __is, __gnu_cxx::von_mises_distribution< _RealType > &__x)

- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **__gnu_cxx::operator>>** (std::basic_istream< _CharT, _Traits > &↩
  __is, arcsine_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **__gnu_cxx::operator>>** (std::basic_istream< _CharT, _Traits > &↩
  __is, hoyt_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **__gnu_cxx::operator>>** (std::basic_istream< _CharT, _Traits > &↩
  __is, k_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **__gnu_cxx::operator>>** (std::basic_istream< _CharT, _Traits > &↩
  __is, logistic_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **__gnu_cxx::operator>>** (std::basic_istream< _CharT, _Traits > &↩
  __is, nakagami_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **__gnu_cxx::operator>>** (std::basic_istream< _CharT, _Traits > &↩
  __is, pareto_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **__gnu_cxx::operator>>** (std::basic_istream< _CharT, _Traits > &↩
  __is, rice_distribution< _RealType > &__x)

6.113.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ext/random>.

6.114 range_access.h File Reference

Namespaces

- namespace std

Functions

- template<typename _Container>
  constexpr auto std::begin (_Container &__cont) noexcept(noexcept(__cont.begin())) -> decltype(__cont.begin())
- template<typename _Tp, size_t _Nm>
  constexpr _Tp * std::begin (_Tp(&__arr)[_Nm]) noexcept
- template<typename _Container>
  constexpr auto std::begin (const _Container &__cont) noexcept(noexcept(__cont.begin())) -> decltype(__cont.↩
  begin())
- template<class _Tp>
  const _Tp * std::begin (const valarray< _Tp > &__va) noexcept
- template<class _Tp>
  _Tp * std::begin (valarray< _Tp > &__va) noexcept
- template<typename _Container>
  constexpr auto std::cbegin (const _Container &__cont) noexcept(noexcept(std::begin(__cont))) -> decltype(std::begin(↩
  __cont))
- template<typename _Container>
  constexpr auto std::cend (const _Container &__cont) noexcept(noexcept(std::end(__cont))) -> decltype(std::end(↩
  __cont))

- template<typename _Container>
  constexpr auto std::crbegin (const _Container &__cont) noexcept(noexcept(std::rbegin(__cont))) -> decltype(std::rbegin(↩ __cont))
- template<typename _Container>
  constexpr auto std::crend (const _Container &__cont) noexcept(noexcept(std::rend(__cont))) -> decltype(std::rend(↩ __cont))
- template<typename _Container>
  constexpr auto std::data (_Container &__cont) noexcept(noexcept(__cont.data())) -> decltype(__cont.data())
- template<typename _Tp, size_t _Nm>
  constexpr _Tp * std::data (_Tp(&__array)[_Nm]) noexcept
- template<typename _Container>
  constexpr auto std::data (const _Container &__cont) noexcept(noexcept(__cont.data())) -> decltype(__cont.↩ data())
- template<typename _Tp>
  constexpr const _Tp * std::data (initializer_list< _Tp > __il) noexcept
- template<typename _Container>
  constexpr auto std::empty (const _Container &__cont) noexcept(noexcept(__cont.empty())) -> decltype(__↩ cont.empty())
- template<typename _Tp, size_t _Nm>
  constexpr bool std::empty (const _Tp(&)[_Nm]) noexcept
- template<typename _Tp>
  constexpr bool std::empty (initializer_list< _Tp > __il) noexcept
- template<typename _Container>
  constexpr auto std::end (_Container &__cont) noexcept(noexcept(__cont.end())) -> decltype(__cont.end())
- template<typename _Tp, size_t _Nm>
  constexpr _Tp * std::end (_Tp(&__arr)[_Nm]) noexcept
- template<typename _Container>
  constexpr auto std::end (const _Container &__cont) noexcept(noexcept(__cont.end())) -> decltype(__cont.end())
- template<class _Tp>
  const _Tp * std::end (const valarray< _Tp > &__va) noexcept
- template<class _Tp>
  _Tp * std::end (valarray< _Tp > &__va) noexcept
- template<typename _Container>
  constexpr auto std::rbegin (_Container &__cont) noexcept(noexcept(__cont.rbegin())) -> decltype(__cont.↩ rbegin())
- template<typename _Tp, size_t _Nm>
  constexpr reverse_iterator< _Tp * > std::rbegin (_Tp(&__arr)[_Nm]) noexcept
- template<typename _Container>
  constexpr auto std::rbegin (const _Container &__cont) noexcept(noexcept(__cont.rbegin())) -> decltype(__↩ cont.rbegin())
- template<typename _Tp>
  constexpr reverse_iterator< const _Tp * > std::rbegin (initializer_list< _Tp > __il) noexcept
- template<typename _Container>
  constexpr auto std::rend (_Container &__cont) noexcept(noexcept(__cont.rend())) -> decltype(__cont.rend())
- template<typename _Tp, size_t _Nm>
  constexpr reverse_iterator< _Tp * > std::rend (_Tp(&__arr)[_Nm]) noexcept
- template<typename _Container>
  constexpr auto std::rend (const _Container &__cont) noexcept(noexcept(__cont.rend())) -> decltype(__cont.↩ rend())
- template<typename _Tp>
  constexpr reverse_iterator< const _Tp * > std::rend (initializer_list< _Tp > __il) noexcept
- template<typename _Container>
  constexpr auto std::size (const _Container &__cont) noexcept(noexcept(__cont.size())) -> decltype(__cont.size())

- template<typename _Tp, size_t _Nm>
  constexpr size_t std::size (const _Tp(&)[_Nm]) noexcept
- template<typename _Container>
  constexpr auto **std::ssize** (const _Container &__cont) noexcept(noexcept(__cont.size())) -> common_type_t< ptrdiff_t, make_signed_t< decltype(__cont.size())> >
- template<typename _Tp, ptrdiff_t _Num>
  constexpr ptrdiff_t **std::ssize** (const _Tp(&)[_Num]) noexcept

6.114.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iterator>.

6.115 ranges_algo.h File Reference

Namespaces

- namespace std

Typedefs

- template<typename _Fp>
  using **std::ranges::__detail::__by_ref_or_value_fn**
- template<typename _Iter1, typename _Iter2, typename _Out>
  using **std::ranges::binary_transform_result**
- template<typename _Iter, typename _Out>
  using **std::ranges::copy_if_result**
- template<typename _Iter, typename _Fp>
  using **std::ranges::for_each_n_result**
- template<typename _Iter, typename _Fp>
  using **std::ranges::for_each_result**
- template<typename _Iter1, typename _Iter2, typename _Out>
  using **std::ranges::merge_result**
- template<typename _Iter>
  using **std::ranges::minmax_element_result**
- template<typename _Tp>
  using **std::ranges::minmax_result**
- template<typename _Iter>
  using **std::ranges::next_permutation_result**
- template<typename _Iter, typename _Out>
  using **std::ranges::partial_sort_copy_result**
- template<typename _Iter, typename _Out1, typename _Out2>
  using **std::ranges::partition_copy_result**
- template<typename _Iter>
  using **std::ranges::prev_permutation_result**
- template<typename _Iter, typename _Out>
  using **std::ranges::remove_copy_if_result**
- template<typename _Iter, typename _Out>
  using **std::ranges::remove_copy_result**
- template<typename _Iter, typename _Out>
  using **std::ranges::replace_copy_if_result**
- template<typename _Iter, typename _Out>
  using **std::ranges::replace_copy_result**

- template<typename _Iter, typename _Out>
  using **std::ranges::reverse_copy_result**
- template<typename _Iter, typename _Out>
  using **std::ranges::rotate_copy_result**
- template<typename _Iter, typename _Out>
  using **std::ranges::set_difference_result**
- template<typename _Iter1, typename _Iter2, typename _Out>
  using **std::ranges::set_intersection_result**
- template<typename _Iter1, typename _Iter2, typename _Out>
  using **std::ranges::set_symmetric_difference_result**
- template<typename _Iter1, typename _Iter2, typename _Out>
  using **std::ranges::set_union_result**
- template<typename _Iter1, typename _Iter2>
  using **std::ranges::swap_ranges_result**
- template<typename _Iter, typename _Out>
  using **std::ranges::unary_transform_result**
- template<typename _Iter, typename _Out>
  using **std::ranges::unique_copy_result**

Functions

- template<typename _Iter, typename _Comp>
  constexpr void **std::ranges::__detail::__adjust_heap** (_Iter __first, iter_difference_t< _Iter > __holeIndex, iter_difference_t< _Iter > __len, iter_value_t< _Iter > __value, _Comp __comp)
- template<typename _Iter, typename _Distance, typename _Compare>
  constexpr void **std::ranges::__detail::__chunk_insertion_sort** (_Iter __first, _Iter __last, _Distance __chunk↩
  _size, _Compare __comp)
- template<typename _Iter, typename _Comp>
  constexpr void **std::ranges::__detail::__final_insertion_sort** (_Iter __first, _Iter __last, _Comp __comp)
- template<typename _Iter, typename _Pred, typename _Distance>
  constexpr _Iter **std::ranges::__detail::__find_if_not_n** (_Iter __first, _Distance &__len, _Pred __pred)
- template<typename _Iter, typename _Comp>
  constexpr void **std::ranges::__detail::__heap_select** (_Iter __first, _Iter __middle, _Iter __last, _Comp __comp)
- template<typename _Iter, typename _Comp>
  constexpr void **std::ranges::__detail::__inplace_stable_sort** (_Iter __first, _Iter __last, _Comp __comp)
- template<typename _Iter, typename _Comp>
  constexpr void **std::ranges::__detail::__insertion_sort** (_Iter __first, _Iter __last, _Comp __comp)
- template<typename _Iter, typename _Comp>
  constexpr void **std::ranges::__detail::__introselect** (_Iter __first, _Iter __nth, _Iter __last, iter_difference_t< _Iter > __depth_limit, _Comp __comp)
- template<typename _Iter, typename _Comp>
  constexpr void **std::ranges::__detail::__introsort_loop** (_Iter __first, _Iter __last, unsigned __depth_limit, _↩
  Comp __comp)
- template<typename _Comp, typename _Proj>
  constexpr _Comp_proj< _Comp, _Proj > **std::ranges::__detail::__make_comp_proj** (_Comp &__comp, _Proj &__proj)
- template<typename _Pred, typename _Proj>
  constexpr _Pred_proj< _Pred, _Proj > **std::ranges::__detail::__make_pred_proj** (_Pred &__pred, _Proj &_↩
  _proj)
- template<typename _Iter, typename _Pointer, typename _Comp>
  void **std::ranges::__detail::__merge_adaptive** (_Iter __first, _Iter __middle, _Iter __last, iter_difference_t< _Iter > __len1, iter_difference_t< _Iter > __len2, _Pointer __buffer, _Comp __comp)

- template<typename _Iter, typename _Distance, typename _Pointer, typename _Comp>
  void **std::ranges::__detail::__merge_adaptive_resize** (_Iter __first, _Iter __middle, _Iter __last, _Distance _↩
  _len1, _Distance __len2, _Pointer __buffer, _Distance __buffer_size, _Comp __comp)
- template<typename _Iter, typename _Out, typename _Distance, typename _Comp>
  void **std::ranges::__detail::__merge_sort_loop** (_Iter __first, _Iter __last, _Out __result, _Distance __step_↩
  size, _Comp __comp)
- template<typename _Iter, typename _Pointer, typename _Comp>
  void **std::ranges::__detail::__merge_sort_with_buffer** (_Iter __first, _Iter __last, _Pointer __buffer, _Comp ↩
  __comp)
- template<typename _Iter, typename _Distance, typename _Comp>
  constexpr void **std::ranges::__detail::__merge_without_buffer** (_Iter __first, _Iter __middle, _Iter __last, _↩
  Distance __len1, _Distance __len2, _Comp __comp)
- template<typename _Iter, typename _Comp>
  constexpr void **std::ranges::__detail::__move_median_to_first** (_Iter __result, _Iter __a, _Iter __b, _Iter __c, _Comp __comp)
- template<typename _Iter, typename _Out, typename _Comp>
  _Out **std::ranges::__detail::__move_merge** (_Iter __first1, _Iter __last1, _Iter __first2, _Iter __last2, _Out __↩
  result, _Comp __comp)
- template<typename _Iter1, typename _Iter2, typename _Out, typename _Comp>
  void **std::ranges::__detail::__move_merge_adaptive** (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Iter2 _↩
  _last2, _Out __result, _Comp __comp)
- template<typename _Iter1, typename _Iter2, typename _Iter3, typename _Comp>
  void **std::ranges::__detail::__move_merge_adaptive_backward** (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Iter2 __last2, _Iter3 __result, _Comp __comp)
- template<typename _Iter, typename _Comp>
  constexpr void **std::ranges::__detail::__partial_sort** (_Iter __first, _Iter __middle, _Iter __last, _Comp __comp)
- template<typename _Iter, typename _Comp>
  constexpr void **std::ranges::__detail::__pop_heap** (_Iter __first, _Iter __last, _Iter __result, _Comp __comp)
- template<typename _Iter, typename _Comp>
  constexpr void **std::ranges::__detail::__push_heap** (_Iter __first, iter_difference_t< _Iter > __holeIndex, iter↩
  _difference_t< _Iter > __topIndex, iter_value_t< _Iter > __value, _Comp __comp)
- template<typename _Iter1, typename _Iter2>
  _Iter1 **std::ranges::__detail::__rotate_adaptive** (_Iter1 __first, _Iter1 __middle, _Iter1 __last, iter_difference_t< _Iter1 > __len1, iter_difference_t< _Iter1 > __len2, _Iter2 __buffer, iter_difference_t< _Iter1 > __buffer_size)
- template<typename _Iter, typename _Sent, typename _Pointer, typename _Pred, typename _Distance>
  constexpr subrange< _Iter > **std::ranges::__detail::__stable_partition_adaptive** (_Iter __first, _Sent __last, _Pred __pred, _Distance __len, _Pointer __buffer, _Distance __buffer_size)
- template<typename _Iter, typename _Pointer, typename _Comp>
  void **std::ranges::__detail::__stable_sort_adaptive** (_Iter __first, _Iter __middle, _Iter __last, _Pointer __buffer, _Comp __comp)
- template<typename _Iter, typename _Pointer, typename _Distance, typename _Comp>
  void **std::ranges::__detail::__stable_sort_adaptive_resize** (_Iter __first, _Iter __last, _Pointer __buffer, _↩
  Distance __buffer_size, _Comp __comp)
- template<typename _Iter, typename _Comp>
  constexpr void **std::ranges::__detail::__unguarded_insertion_sort** (_Iter __first, _Iter __last, _Comp __comp)
- template<typename _Iter, typename _Comp>
  constexpr void **std::ranges::__detail::__unguarded_linear_insert** (_Iter __last, _Comp __comp)
- template<typename _Iter, typename _Comp>
  constexpr _Iter **std::ranges::__detail::__unguarded_partition** (_Iter __first, _Iter __last, _Iter __pivot, _Comp __comp)
- template<typename _Iter, typename _Comp>
  constexpr _Iter **std::ranges::__detail::__unguarded_partition_pivot** (_Iter __first, _Iter __last, _Comp __↩
  comp)

- template<typename _ForwardIterator>
  constexpr _ForwardIterator **std::shift_left** (_ForwardIterator __first, _ForwardIterator __last, typename iterator_traits< _ForwardIterator >::difference_type __n)
- template<typename _ForwardIterator>
  constexpr _ForwardIterator **std::shift_right** (_ForwardIterator __first, _ForwardIterator __last, typename iterator_traits< _ForwardIterator >::difference_type __n)

Variables

- constexpr int **std::ranges::__detail::__sort_threshold**
- constexpr __all_of_fn **std::ranges::all_of**
- constexpr __any_of_fn **std::ranges::any_of**
- constexpr __binary_search_fn **std::ranges::binary_search**
- constexpr __clamp_fn **std::ranges::clamp**
- constexpr __copy_if_fn **std::ranges::copy_if**
- constexpr __count_fn **std::ranges::count**
- constexpr __count_if_fn **std::ranges::count_if**
- constexpr __equal_range_fn **std::ranges::equal_range**
- constexpr __find_end_fn **std::ranges::find_end**
- constexpr __find_first_of_fn **std::ranges::find_first_of**
- constexpr __for_each_fn **std::ranges::for_each**
- constexpr __for_each_n_fn **std::ranges::for_each_n**
- constexpr __generate_fn **std::ranges::generate**
- constexpr __generate_n_fn **std::ranges::generate_n**
- constexpr __includes_fn **std::ranges::includes**
- constexpr __inplace_merge_fn **std::ranges::inplace_merge**
- constexpr __is_heap_fn **std::ranges::is_heap**
- constexpr __is_heap_until_fn **std::ranges::is_heap_until**
- constexpr __is_partitioned_fn **std::ranges::is_partitioned**
- constexpr __is_permutation_fn **std::ranges::is_permutation**
- constexpr __is_sorted_fn **std::ranges::is_sorted**
- constexpr __is_sorted_until_fn **std::ranges::is_sorted_until**
- constexpr __lexicographical_compare_fn **std::ranges::lexicographical_compare**
- constexpr __lower_bound_fn **std::ranges::lower_bound**
- constexpr __make_heap_fn **std::ranges::make_heap**
- constexpr __max_fn **std::ranges::max**
- constexpr __max_element_fn **std::ranges::max_element**
- constexpr __merge_fn **std::ranges::merge**
- constexpr __min_element_fn **std::ranges::min_element**
- constexpr __minmax_fn **std::ranges::minmax**
- constexpr __minmax_element_fn **std::ranges::minmax_element**
- constexpr __next_permutation_fn **std::ranges::next_permutation**
- constexpr __none_of_fn **std::ranges::none_of**
- constexpr __nth_element_fn **std::ranges::nth_element**
- constexpr __partial_sort_fn **std::ranges::partial_sort**
- constexpr __partial_sort_copy_fn **std::ranges::partial_sort_copy**
- constexpr __partition_fn **std::ranges::partition**
- constexpr __partition_copy_fn **std::ranges::partition_copy**
- constexpr __partition_point_fn **std::ranges::partition_point**
- constexpr __pop_heap_fn **std::ranges::pop_heap**
- constexpr __prev_permutation_fn **std::ranges::prev_permutation**

- constexpr __push_heap_fn **std::ranges::push_heap**
- constexpr __remove_fn **std::ranges::remove**
- constexpr __remove_copy_fn **std::ranges::remove_copy**
- constexpr __remove_copy_if_fn **std::ranges::remove_copy_if**
- constexpr __remove_if_fn **std::ranges::remove_if**
- constexpr __replace_fn **std::ranges::replace**
- constexpr __replace_copy_fn **std::ranges::replace_copy**
- constexpr __replace_copy_if_fn **std::ranges::replace_copy_if**
- constexpr __replace_if_fn **std::ranges::replace_if**
- constexpr __reverse_fn **std::ranges::reverse**
- constexpr __reverse_copy_fn **std::ranges::reverse_copy**
- constexpr __rotate_fn **std::ranges::rotate**
- constexpr __rotate_copy_fn **std::ranges::rotate_copy**
- constexpr __sample_fn **std::ranges::sample**
- constexpr __search_n_fn **std::ranges::search_n**
- constexpr __set_difference_fn **std::ranges::set_difference**
- constexpr __set_intersection_fn **std::ranges::set_intersection**
- constexpr __set_symmetric_difference_fn **std::ranges::set_symmetric_difference**
- constexpr __set_union_fn **std::ranges::set_union**
- constexpr __shuffle_fn **std::ranges::shuffle**
- constexpr __sort_fn **std::ranges::sort**
- constexpr __sort_heap_fn **std::ranges::sort_heap**
- constexpr __stable_partition_fn **std::ranges::stable_partition**
- constexpr __stable_sort_fn **std::ranges::stable_sort**
- constexpr __swap_ranges_fn **std::ranges::swap_ranges**
- constexpr __transform_fn **std::ranges::transform**
- constexpr __unique_fn **std::ranges::unique**
- constexpr __unique_copy_fn **std::ranges::unique_copy**
- constexpr __upper_bound_fn **std::ranges::upper_bound**

6.115.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <algorithm>.

6.116 ranges_algobase.h File Reference

6.116.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <algorithm>.

6.117 ranges_base.h File Reference

Macros

- #define **_GLIBCXX26_RANGE_ALGO_DEF_VAL_T**(_I, _P)

6.117.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ranges>.

6.118 ranges_cmp.h File Reference

Classes

- struct std::ranges::equal_to
- struct std::ranges::greater
- struct std::ranges::greater_equal
- struct std::identity
- struct std::ranges::less
- struct std::ranges::less_equal
- struct std::ranges::not_equal_to

Namespaces

- namespace std

6.118.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <functional>.

6.119 ranges_uninitialized.h File Reference

6.119.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

6.120 ranges_util.h File Reference

Classes

- class std::ranges::subrange< _It, _Sent, _Kind >
- class std::ranges::view_interface< _Derived >

Namespaces

- namespace std

Typedefs

- template<range _Range>
  using **std::ranges::borrowed_subrange_t**
- template<typename _Iter1, typename _Iter2>
  using **std::ranges::mismatch_result**

Enumerations

- enum class **subrange_kind** : bool { **unsized** , **sized** }

Functions

- template<size_t _Num, class _It, class _Sent, subrange_kind _Kind>
  requires ((_Num == 0 && copyable<_It>) || _Num == 1)
  constexpr auto **std::get** (const subrange< _It, _Sent, _Kind > &__r)

- template<size_t _Num, class _It, class _Sent, subrange_kind _Kind>
  requires ((_Num == 0 && copyable<_It>) || _Num == 1)
  constexpr auto **std::ranges::get** (const subrange< _It, _Sent, _Kind > &__r)
- template<size_t _Num, class _It, class _Sent, subrange_kind _Kind>
  requires (_Num < 2)
  constexpr auto **std::ranges::get** (subrange< _It, _Sent, _Kind > &&__r)
- template<input_or_output_iterator _It, sentinel_for< _It > _Sent>
  std::ranges::subrange (_It, _Sent) -> subrange< _It, _Sent >
- template<input_or_output_iterator _It, sentinel_for< _It > _Sent>
  std::ranges::subrange (_It, _Sent, __detail::__make_unsigned_like_t< iter_difference_t< _It > >) -> subrange< _It, _Sent, subrange_kind::sized >
- template<borrowed_range _Rng>
  std::ranges::subrange (_Rng &&) -> subrange< iterator_t< _Rng >, sentinel_t< _Rng >,(sized_range< _Rng >||sized_sentinel_for< sentinel_t< _Rng >, iterator_t< _Rng > >) ? subrange_kind::sized :subrange_kind↩ ::unsized >
- template<borrowed_range _Rng>
  std::ranges::subrange (_Rng &&, __detail::__make_unsigned_like_t< range_difference_t< _Rng > >) -> subrange< iterator_t< _Rng >, sentinel_t< _Rng >, subrange_kind::sized >

Variables

- constexpr __adjacent_find_fn **std::ranges::adjacent_find**
- template<typename _It, typename _Sent, subrange_kind _Kind>
  constexpr bool **std::ranges::enable_borrowed_range**< **subrange**< **_It, _Sent, _Kind** > >
- constexpr __find_fn **std::ranges::find**
- constexpr __find_if_fn **std::ranges::find_if**
- constexpr __find_if_not_fn **std::ranges::find_if_not**
- constexpr __min_fn **std::ranges::min**
- constexpr __mismatch_fn **std::ranges::mismatch**
- constexpr __search_fn **std::ranges::search**

6.120.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ranges>.

6.121 refwrap.h File Reference

Classes

- class std::reference_wrapper< _Tp >

Namespaces

- namespace std

Functions

- template<typename _Tp>
  std::reference_wrapper (_Tp &) -> reference_wrapper< _Tp >

6.121.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <functional>.

6.122 regex.h File Reference

Classes

- class std::basic_regex< _Ch_type, _Rx_traits >
- class std::match_results< _Bi_iter, _Alloc >
- class std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >
- class std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >
- class std::regex_traits< _Ch_type >
- class std::sub_match< _BiIter >

Namespaces

- namespace std
- namespace std::__detail

Typedefs

- typedef match_results< const char * > **std::cmatch**
- typedef regex_iterator< const char * > **std::cregex_iterator**
- typedef regex_token_iterator< const char * > std::cregex_token_iterator
- typedef sub_match< const char * > std::csub_match
- typedef basic_regex< char > std::regex
- typedef match_results< string::const_iterator > **std::smatch**
- typedef regex_iterator< string::const_iterator > **std::sregex_iterator**
- typedef regex_token_iterator< string::const_iterator > std::sregex_token_iterator
- typedef sub_match< string::const_iterator > std::ssub_match
- typedef match_results< const wchar_t * > **std::wcmatch**
- typedef regex_iterator< const wchar_t * > **std::wcregex_iterator**
- typedef regex_token_iterator< const wchar_t * > std::wcregex_token_iterator
- typedef sub_match< const wchar_t * > std::wcsub_match
- typedef basic_regex< wchar_t > std::wregex
- typedef match_results< wstring::const_iterator > **std::wsmatch**
- typedef regex_iterator< wstring::const_iterator > **std::wsregex_iterator**
- typedef regex_token_iterator< wstring::const_iterator > std::wsregex_token_iterator
- typedef sub_match< wstring::const_iterator > std::wssub_match

Enumerations

- enum class **_RegexExecutorPolicy** : int { **_S_auto** , **_S_alternate** }

Functions

- template<typename _BiIter, typename _Alloc, typename _CharT, typename _TraitsT>
  bool **std::__detail::__regex_algo_impl** (_BiIter __s, _BiIter __e, match_results< _BiIter, _Alloc > &__m, const basic_regex< _CharT, _TraitsT > &__re, regex_constants::match_flag_type __flags, _RegexExecutorPolicy _↩
  _policy, bool __match_mode)
- template<typename _ForwardIterator>
  std::basic_regex (_ForwardIterator, _ForwardIterator, regex_constants::syntax_option_type={}) -> basic_↩
  regex< typename iterator_traits< _ForwardIterator >::value_type >

Matching, Searching, and Replacing

- template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>
  bool std::regex_match (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits>
  bool std::regex_match (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Ch_type, class _Rx_traits>
  bool std::regex_match (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_constants::match_default)
- template<typename _Ch_type, typename _Alloc, typename _Rx_traits>
  bool std::regex_match (const _Ch_type *__s, match_results< const _Ch_type *, _Alloc > &↩ __m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __↩ f=regex_constants::match_default)
- template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>
  bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &&, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &, const basic_regex< _Ch_type, _Rx_traits > &, regex_constants::match_flag_type=regex_constants::match_default)=delete
- template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>
  bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Ch_traits, typename _Str_allocator, typename _Ch_type, typename _Rx_traits>
  bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Str_allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type>
  _Out_iter std::regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__e, const _Ch_type *__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa>
  _Out_iter std::regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__e, const basic_string< _Ch_type, _St, _Sa > &__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Rx_traits, typename _Ch_type>
  basic_string< _Ch_type > std::regex_replace (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_↩ traits > &__e, const _Ch_type *__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa>
  basic_string< _Ch_type > std::regex_replace (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx↩ _traits > &__e, const basic_string< _Ch_type, _St, _Sa > &__fmt, regex_constants::match_flag_type __↩ flags=regex_constants::match_default)
- template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa>
  basic_string< _Ch_type, _St, _Sa > std::regex_replace (const basic_string< _Ch_type, _St, _Sa > &__s, const basic_regex< _Ch_type, _Rx_traits > &__e, const _Ch_type *__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa, typename _Fst, typename _Fsa>
  basic_string< _Ch_type, _St, _Sa > std::regex_replace (const basic_string< _Ch_type, _St, _Sa > &__↩ s, const basic_regex< _Ch_type, _Rx_traits > &__e, const basic_string< _Ch_type, _Fst, _Fsa > &__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>
  bool std::regex_search (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits>
  bool std::regex_search (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Ch_type, typename _Rx_traits>
  bool std::regex_search (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits > &__↩ e, regex_constants::match_flag_type __f=regex_constants::match_default)

- template<typename _Ch_type, class _Alloc, class _Rx_traits>
  bool std::regex_search (const _Ch_type *__s, match_results< const _Ch_type *, _Alloc > &↩
  __m, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __↩
  f=regex_constants::match_default)
- template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>
  bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &&, match_results< type-
  name basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &, const basic_regex< _↩
  Ch_type, _Rx_traits > &, regex_constants::match_flag_type=regex_constants::match_default)=delete
- template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>
  bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< type-
  name basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex<
  _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)
- template<typename _Ch_traits, typename _String_allocator, typename _Ch_type, typename _Rx_traits>
  bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _String_allocator > &__s, const
  basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __flags=regex_constants::match_default)

6.122.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <regex>.

6.123 regex.tcc File Reference

Namespaces

- namespace std
- namespace std::__detail

Functions

- template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type>
  _Out_iter **std::__regex_replace** (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type,
  _Rx_traits > &__e, const _Ch_type *__fmt, size_t __len, regex_constants::match_flag_type __flags)

6.123.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <regex>.

6.124 regex_automaton.h File Reference

Classes

- class std::__detail::_StateSeq< _TraitsT >

Namespaces

- namespace std
- namespace std::__detail

Macros

- #define **_GLIBCXX_REGEX_STATE_LIMIT**

Typedefs

- template<typename _CharT>
  using **std::__detail::_Matcher**
- typedef long **std::__detail::_StateIdT**

Enumerations

- enum std::__detail::_Opcode : int {
  _S_opcode_unknown , **_S_opcode_alternative** , **_S_opcode_repeat** , **_S_opcode_backref** ,
  _S_opcode_line_begin_assertion , **_S_opcode_line_end_assertion** , **_S_opcode_word_boundary** , **_S_↩opcode_subexpr_lookahead** ,
  _S_opcode_subexpr_begin , **_S_opcode_subexpr_end** , **_S_opcode_dummy** , **_S_opcode_match** ,
  _S_opcode_accept }

Variables

- constexpr _StateIdT **std::__detail::_S_invalid_state_id**

6.124.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <regex>.

6.125 regex_automaton.tcc File Reference

Namespaces

- namespace std
- namespace std::__detail

6.125.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <regex>.

6.126 regex_compiler.h File Reference

Classes

- struct std::__detail::_BracketMatcher< _TraitsT, __icase, __collate >
- class std::__detail::_Compiler< _TraitsT >

Namespaces

- namespace std
- namespace std::__detail

6.126.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <regex>.

6.127 regex_compiler.tcc File Reference

Namespaces

- namespace std
- namespace std::__detail

Macros

- #define **__INSERT_REGEX_MATCHER**(__func, ...)

6.127.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <regex>.

6.128 regex_constants.h File Reference

Namespaces

- namespace std
- namespace std::regex_constants

5.1 Regular Expression Syntax Options

- constexpr syntax_option_type std::regex_constants::__multiline
- constexpr syntax_option_type std::regex_constants::__polynomial
- constexpr syntax_option_type std::regex_constants::awk
- constexpr syntax_option_type std::regex_constants::basic
- constexpr syntax_option_type std::regex_constants::collate
- constexpr syntax_option_type std::regex_constants::ECMAScript
- constexpr syntax_option_type std::regex_constants::egrep
- constexpr syntax_option_type std::regex_constants::extended
- constexpr syntax_option_type std::regex_constants::grep
- constexpr syntax_option_type std::regex_constants::icase
- constexpr syntax_option_type std::regex_constants::multiline
- constexpr syntax_option_type std::regex_constants::nosubs
- constexpr syntax_option_type std::regex_constants::operator& (syntax_option_type __a, syntax_option_type _↩ _b) noexcept
- constexpr syntax_option_type & std::regex_constants::operator&= (syntax_option_type &__a, syntax_option_type __b) noexcept
- constexpr syntax_option_type std::regex_constants::operator^ (syntax_option_type __a, syntax_option_type _↩ _b) noexcept
- constexpr syntax_option_type & std::regex_constants::operator^= (syntax_option_type &__a, syntax_option_type __b) noexcept
- constexpr syntax_option_type std::regex_constants::operator| (syntax_option_type __a, syntax_option_type __b) noexcept
- constexpr syntax_option_type & std::regex_constants::operator|= (syntax_option_type &__a, syntax_option_type __b) noexcept
- constexpr syntax_option_type std::regex_constants::operator~ (syntax_option_type __a) noexcept
- constexpr syntax_option_type std::regex_constants::optimize
- enum std::regex_constants::syntax_option_type : unsigned int {
  _S_icase , **_S_nosubs** , **_S_optimize** , **_S_collate** ,
  _S_ECMAScript , **_S_basic** , **_S_extended** , **_S_awk** ,
  _S_grep , **_S_egrep** , **_S_polynomial** , **_S_multiline** }

5.2 Matching Rules

Matching a regular expression against a sequence of characters [first, last) proceeds according to the rules of the grammar specified for the regular expression object, modified according to the effects listed below for any bitmask elements set.

- constexpr match_flag_type std::regex_constants::format_default
- constexpr match_flag_type std::regex_constants::format_first_only
- constexpr match_flag_type std::regex_constants::format_no_copy
- constexpr match_flag_type std::regex_constants::format_sed
- constexpr match_flag_type std::regex_constants::match_any
- constexpr match_flag_type std::regex_constants::match_continuous
- constexpr match_flag_type std::regex_constants::match_default
- enum std::regex_constants::match_flag_type : unsigned int { **_S_default** , **_S_not_bol** , **_S_not_eol** , **_S_not_bow** , **_S_not_eow** , **_S_any** , **_S_not_null** , **_S_continuous** , **_S_prev_avail** , **_S_sed** , **_S_no_copy** , **_S_first_only** , **_S_match_flag_last** }
- constexpr match_flag_type std::regex_constants::match_not_bol
- constexpr match_flag_type std::regex_constants::match_not_bow
- constexpr match_flag_type std::regex_constants::match_not_eol
- constexpr match_flag_type std::regex_constants::match_not_eow
- constexpr match_flag_type std::regex_constants::match_not_null
- constexpr match_flag_type std::regex_constants::match_prev_avail
- constexpr match_flag_type std::regex_constants::operator& (match_flag_type __a, match_flag_type __b) noexcept
- constexpr match_flag_type & std::regex_constants::operator&= (match_flag_type &__a, match_flag_type __b) noexcept
- constexpr match_flag_type std::regex_constants::operator^ (match_flag_type __a, match_flag_type __b) noexcept
- constexpr match_flag_type & std::regex_constants::operator^= (match_flag_type &__a, match_flag_type __b) noexcept
- constexpr match_flag_type std::regex_constants::operator| (match_flag_type __a, match_flag_type __b) noexcept
- constexpr match_flag_type & std::regex_constants::operator|= (match_flag_type &__a, match_flag_type __b) noexcept
- constexpr match_flag_type std::regex_constants::operator~ (match_flag_type __a) noexcept

6.128.1 Detailed Description

Constant definitions for the std regex library.
This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <regex>.

6.129 regex_error.h File Reference

Classes

- class std::regex_error

Namespaces

- namespace std
- namespace std::regex_constants

5.3 Error Types

- constexpr error_type std::regex_constants::error_backref (_S_error_backref)
- constexpr error_type std::regex_constants::error_badbrace (_S_error_badbrace)
- constexpr error_type std::regex_constants::error_badrepeat (_S_error_badrepeat)
- constexpr error_type std::regex_constants::error_brace (_S_error_brace)
- constexpr error_type std::regex_constants::error_brack (_S_error_brack)
- constexpr error_type std::regex_constants::error_collate (_S_error_collate)
- constexpr error_type std::regex_constants::error_complexity (_S_error_complexity)
- constexpr error_type std::regex_constants::error_ctype (_S_error_ctype)
- constexpr error_type std::regex_constants::error_escape (_S_error_escape)
- constexpr error_type std::regex_constants::error_paren (_S_error_paren)
- constexpr error_type std::regex_constants::error_range (_S_error_range)
- constexpr error_type std::regex_constants::error_space (_S_error_space)
- constexpr error_type std::regex_constants::error_stack (_S_error_stack)
- enum std::regex_constants::error_type {
  _S_error_collate , **_S_error_ctype** , **_S_error_escape** , **_S_error_backref** ,
  _S_error_brack , **_S_error_paren** , **_S_error_brace** , **_S_error_badbrace** ,
  _S_error_range , **_S_error_space** , **_S_error_badrepeat** , **_S_error_complexity** ,
  _S_error_stack , **_S_null** , **_S_grammar** }

6.129.1 Detailed Description

Error and exception objects for the std regex library.
This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <regex>.

6.130 regex_executor.h File Reference

Classes

- class std::__detail::_Executor< _BiIter, _Alloc, _TraitsT, __dfs_mode >

Namespaces

- namespace std
- namespace std::__detail

6.130.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <regex>.

6.131 regex_executor.tcc File Reference

Namespaces

- namespace std
- namespace std::__detail

6.131.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <regex>.

6.132 regex_scanner.h File Reference

Classes

- class std::__detail::_Scanner< _CharT >

Namespaces

- namespace std
- namespace std::__detail

6.132.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <regex>.

6.133 regex_scanner.tcc File Reference

Namespaces

- namespace std
- namespace std::__detail

6.133.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <regex>.

6.134 requires_hosted.h File Reference

6.134.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <version>.

6.135 sat_arith.h File Reference

6.135.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <numeric>.

6.136 semaphore_base.h File Reference

6.136.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <semaphore>.

6.137 bits/shared_ptr.h File Reference

Classes

- class std::enable_shared_from_this< _Tp >
- struct std::hash< shared_ptr< _Tp > >
- struct std::owner_less< shared_ptr< _Tp > >
- struct std::owner_less< void >

- struct std::owner_less< weak_ptr< _Tp > >
- class std::shared_ptr< _Tp >
- class std::weak_ptr< _Tp >

Namespaces

- namespace std
- namespace std::__detail

Functions

- template<typename _Del, typename _Tp, _Lock_policy _Lp>
  _Del * **std::get_deleter** (const __shared_ptr< _Tp, _Lp > &__p) noexcept
- template<typename _Tp, typename _Del>
  std::shared_ptr (unique_ptr< _Tp, _Del >) -> shared_ptr< _Tp >
- template<typename _Tp>
  std::shared_ptr (weak_ptr< _Tp >) -> shared_ptr< _Tp >
- template<typename _Tp>
  std::weak_ptr (shared_ptr< _Tp >) -> weak_ptr< _Tp >

6.137.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

6.138 experimental/bits/shared_ptr.h File Reference

Classes

- struct std::hash< experimental::shared_ptr< _Tp > >
- struct std::experimental::fundamentals_v2::owner_less< shared_ptr< _Tp > >
- struct std::experimental::fundamentals_v2::owner_less< weak_ptr< _Tp > >

Namespaces

- namespace std
- namespace std::experimental

Functions

- template<typename _Tp>
  bool **std::experimental::atomic_compare_exchange_strong** (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w)
- template<typename _Tp>
  bool **std::experimental::atomic_compare_exchange_strong_explicit** (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w, memory_order __success, memory_order __failure)
- template<typename _Tp>
  bool **std::experimental::atomic_compare_exchange_weak** (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w)
- template<typename _Tp>
  bool **std::experimental::atomic_compare_exchange_weak_explicit** (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w, memory_order __success, memory_order __failure)
- template<typename _Tp>
  void **std::experimental::atomic_exchange** (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r)

- template<typename _Tp>
  shared_ptr< _Tp > **std::experimental::atomic_exchange_explicit** (const shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r, memory_order __mo)
- template<typename _Tp>
  bool **std::experimental::atomic_is_lock_free** (const shared_ptr< _Tp > *__p)
- template<typename _Tp>
  shared_ptr< _Tp > **std::experimental::atomic_load** (const shared_ptr< _Tp > *__p)
- template<typename _Tp>
  shared_ptr< _Tp > **std::experimental::atomic_load_explicit** (const shared_ptr< _Tp > *__p, memory_order __mo)
- template<typename _Tp>
  void **std::experimental::atomic_store** (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r)
- template<typename _Tp>
  shared_ptr< _Tp > **std::experimental::atomic_store_explicit** (const shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r, memory_order __mo)
- template<typename _Tp, typename _Tp1>
  shared_ptr< _Tp > **std::experimental::const_pointer_cast** (const shared_ptr< _Tp1 > &__r) noexcept
- template<typename _Tp, typename _Tp1>
  shared_ptr< _Tp > **std::experimental::dynamic_pointer_cast** (const shared_ptr< _Tp1 > &__r) noexcept
- template<typename _Del, typename _Tp>
  _Del * std::experimental::get_deleter (const shared_ptr< _Tp > &__p) noexcept
- template<typename _Tp>
  bool **std::experimental::operator!=** (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp1, typename _Tp2>
  bool **std::experimental::operator!=** (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept
- template<typename _Tp>
  bool **std::experimental::operator!=** (nullptr_t, const shared_ptr< _Tp > &__a) noexcept
- template<typename _Tp>
  bool **std::experimental::operator<** (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp1, typename _Tp2>
  bool **std::experimental::operator<** (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept
- template<typename _Tp>
  bool **std::experimental::operator<** (nullptr_t, const shared_ptr< _Tp > &__a) noexcept
- template<typename _Ch, typename _Tr, typename _Tp>
  std::basic_ostream< _Ch, _Tr > & **std::experimental::operator<<** (std::basic_ostream< _Ch, _Tr > &__os, const shared_ptr< _Tp > &__p)
- template<typename _Tp>
  bool **std::experimental::operator<=** (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp1, typename _Tp2>
  bool **std::experimental::operator<=** (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept
- template<typename _Tp>
  bool **std::experimental::operator<=** (nullptr_t, const shared_ptr< _Tp > &__a) noexcept
- template<typename _Tp>
  bool **std::experimental::operator==** (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp1, typename _Tp2>
  bool **std::experimental::operator==** (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept
- template<typename _Tp>
  bool **std::experimental::operator==** (nullptr_t, const shared_ptr< _Tp > &__a) noexcept

- template<typename _Tp>
  bool **std::experimental::operator>** (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp1, typename _Tp2>
  bool **std::experimental::operator>** (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept
- template<typename _Tp>
  bool **std::experimental::operator>** (nullptr_t, const shared_ptr< _Tp > &__a) noexcept
- template<typename _Tp>
  bool **std::experimental::operator>=** (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp1, typename _Tp2>
  bool **std::experimental::operator>=** (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept
- template<typename _Tp>
  bool **std::experimental::operator>=** (nullptr_t, const shared_ptr< _Tp > &__a) noexcept
- template<typename _Tp, typename _Tp1>
  shared_ptr< _Tp > **std::experimental::reinterpret_pointer_cast** (const shared_ptr< _Tp1 > &__r) noexcept
- template<typename _Tp, typename _Tp1>
  shared_ptr< _Tp > **std::experimental::static_pointer_cast** (const shared_ptr< _Tp1 > &__r) noexcept
- template<typename _Tp>
  void **std::experimental::swap** (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b) noexcept
- template<typename _Tp>
  void **std::experimental::swap** (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b) noexcept

Variables

- template<typename _Yp, typename _Tp>
  constexpr bool **std::experimental::__sp_compatible_v**
- template<typename _Tp, typename _Yp>
  constexpr bool **std::experimental::__sp_is_constructible_v**

6.138.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <experimental/memory>.

6.139 shared_ptr_atomic.h File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_TSAN_MUTEX_DESTROY**(X)
- #define **_GLIBCXX_TSAN_MUTEX_LOCKED**(X)
- #define **_GLIBCXX_TSAN_MUTEX_POST_SIGNAL**(X)
- #define **_GLIBCXX_TSAN_MUTEX_POST_UNLOCK**(X)
- #define **_GLIBCXX_TSAN_MUTEX_PRE_SIGNAL**(X)
- #define **_GLIBCXX_TSAN_MUTEX_PRE_UNLOCK**(X)
- #define **_GLIBCXX_TSAN_MUTEX_TRY_LOCK**(X)
- #define **_GLIBCXX_TSAN_MUTEX_TRY_LOCK_FAILED**(X)

Functions

- template<typename _Tp, _Lock_policy _Lp>
  bool std::atomic_is_lock_free (const __shared_ptr< _Tp, _Lp > *)
- template<typename _Tp>
  bool std::atomic_is_lock_free (const shared_ptr< _Tp > *__p)

- template<typename _Tp, _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > std::atomic_load (const __shared_ptr< _Tp, _Lp > *__p)
- template<typename _Tp>
  shared_ptr< _Tp > std::atomic_load (const shared_ptr< _Tp > *__p)
- template<typename _Tp, _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > std::atomic_load_explicit (const __shared_ptr< _Tp, _Lp > *__p, memory_order)
- template<typename _Tp>
  shared_ptr< _Tp > std::atomic_load_explicit (const shared_ptr< _Tp > *__p, memory_order)

- template<typename _Tp, _Lock_policy _Lp>
  void std::atomic_store (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r)
- template<typename _Tp>
  void std::atomic_store (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r)
- template<typename _Tp, _Lock_policy _Lp>
  void std::atomic_store_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r, memory_order)
- template<typename _Tp>
  void std::atomic_store_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r, memory_order)

- template<typename _Tp, _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > std::atomic_exchange (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r)
- template<typename _Tp>
  shared_ptr< _Tp > std::atomic_exchange (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r)
- template<typename _Tp, _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > std::atomic_exchange_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r, memory_order)
- template<typename _Tp>
  shared_ptr< _Tp > std::atomic_exchange_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r, memory_order)

- template<typename _Tp, _Lock_policy _Lp>
  bool std::atomic_compare_exchange_strong (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w)
- template<typename _Tp>
  bool std::atomic_compare_exchange_strong (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w)
- template<typename _Tp, _Lock_policy _Lp>
  bool std::atomic_compare_exchange_strong_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w, memory_order, memory_order)

- template<typename _Tp>
  bool std::atomic_compare_exchange_strong_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w, memory_order, memory_order)
- template<typename _Tp, _Lock_policy _Lp>
  bool std::atomic_compare_exchange_weak (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w)
- template<typename _Tp>
  bool std::atomic_compare_exchange_weak (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w)
- template<typename _Tp, _Lock_policy _Lp>
  bool std::atomic_compare_exchange_weak_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w, memory_order __success, memory_order __failure)
- template<typename _Tp>
  bool std::atomic_compare_exchange_weak_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w, memory_order __success, memory_order __failure)

6.139.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

6.140 shared_ptr_base.h File Reference

Classes

- struct std::_Sp_ebo_helper< _Nm, _Tp, false >
- struct std::_Sp_ebo_helper< _Nm, _Tp, true >
- class std::bad_weak_ptr
- struct std::hash< __shared_ptr< _Tp, _Lp > >

Namespaces

- namespace std

Enumerations

- enum **_Lock_policy**

Functions

- template<typename _Tp, _Lock_policy _Lp = __default_lock_policy, typename _Alloc, typename... _Args>
  __shared_ptr< _Tp, _Lp > **std::__allocate_shared** (const _Alloc &__a, _Args &&... __args)
- template<typename _Tp, _Lock_policy _Lp = __default_lock_policy, typename... _Args>
  __shared_ptr< _Tp, _Lp > **std::__make_shared** (_Args &&... __args)
- template<typename _Tp>
  _Tp * **std::__shared_ptr_deref** (_Tp *__p)
- void **std::__throw_bad_weak_ptr** ()
- template<typename _Tp, typename _Tp1, _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > std::const_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept
- template<typename _Tp, typename _Tp1, _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > std::dynamic_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept
- template<typename _Tp, typename _Up, _Lock_policy _Lp>
  strong_ordering **std::operator<=>** (const __shared_ptr< _Tp, _Lp > &__a, const __shared_ptr< _Up, _Lp > &__b) noexcept

- template<typename _Tp, _Lock_policy _Lp>
  strong_ordering **std::operator<=>** (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept
- template<typename _Tp, _Lock_policy _Lp>
  bool **std::operator==** (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept
- template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>
  bool **std::operator==** (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept
- template<typename _Tp, typename _Tp1, _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > **std::reinterpret_pointer_cast** (const __shared_ptr< _Tp1, _Lp > &__r) noexcept
- template<typename _Tp, typename _Tp1, _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > std::static_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept
- template<typename _Tp, _Lock_policy _Lp>
  void **std::swap** (__shared_ptr< _Tp, _Lp > &__a, __shared_ptr< _Tp, _Lp > &__b) noexcept
- template<typename _Tp, _Lock_policy _Lp>
  void **std::swap** (__weak_ptr< _Tp, _Lp > &__a, __weak_ptr< _Tp, _Lp > &__b) noexcept

Variables

- const _Lock_policy **std::__default_lock_policy**

6.140.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

6.141 slice_array.h File Reference

Classes

- class std::slice
- class std::slice_array< _Tp >

Namespaces

- namespace std

6.141.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <valarray>.

6.142 specfun.h File Reference

Namespaces

- namespace __gnu_cxx
- namespace std

Functions

- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type __gnu_cxx::airy_ai (_Tp __x)
- float __gnu_cxx::airy_aif (float __x)
- long double __gnu_cxx::airy_ail (long double __x)

- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type __gnu_cxx::airy_bi (_Tp __x)
- float __gnu_cxx::airy_bif (float __x)
- long double __gnu_cxx::airy_bil (long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::assoc_laguerre (unsigned int __n, unsigned int __m, _Tp __x)
- float std::assoc_laguerref (unsigned int __n, unsigned int __m, float __x)
- long double std::assoc_laguerrel (unsigned int __n, unsigned int __m, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::assoc_legendre (unsigned int __l, unsigned int __m, _Tp __x)
- float std::assoc_legendref (unsigned int __l, unsigned int __m, float __x)
- long double std::assoc_legendrel (unsigned int __l, unsigned int __m, long double __x)
- template<typename _Tpa, typename _Tpb>
  __gnu_cxx::__promote_2< _Tpa, _Tpb >::__type std::beta (_Tpa __a, _Tpb __b)
- float std::betaf (float __a, float __b)
- long double std::betal (long double __a, long double __b)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::comp_ellint_1 (_Tp __k)
- float std::comp_ellint_1f (float __k)
- long double std::comp_ellint_1l (long double __k)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::comp_ellint_2 (_Tp __k)
- float std::comp_ellint_2f (float __k)
- long double std::comp_ellint_2l (long double __k)
- template<typename _Tp, typename _Tpn>
  __gnu_cxx::__promote_2< _Tp, _Tpn >::__type std::comp_ellint_3 (_Tp __k, _Tpn __nu)
- float std::comp_ellint_3f (float __k, float __nu)
- long double std::comp_ellint_3l (long double __k, long double __nu)
- template<typename _Tpa, typename _Tpc, typename _Tp>
  __gnu_cxx::__promote_3< _Tpa, _Tpc, _Tp >::__type __gnu_cxx::conf_hyperg (_Tpa __a, _Tpc __c, _Tp __x)
- float __gnu_cxx::conf_hypergf (float __a, float __c, float __x)
- long double __gnu_cxx::conf_hypergl (long double __a, long double __c, long double __x)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_bessel_i (_Tpnu __nu, _Tp __x)
- float std::cyl_bessel_if (float __nu, float __x)
- long double std::cyl_bessel_il (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_bessel_j (_Tpnu __nu, _Tp __x)
- float std::cyl_bessel_jf (float __nu, float __x)
- long double std::cyl_bessel_jl (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_bessel_k (_Tpnu __nu, _Tp __x)
- float std::cyl_bessel_kf (float __nu, float __x)
- long double std::cyl_bessel_kl (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_neumann (_Tpnu __nu, _Tp __x)
- float std::cyl_neumannf (float __nu, float __x)
- long double std::cyl_neumannl (long double __nu, long double __x)
- template<typename _Tp, typename _Tpp>
  __gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::ellint_1 (_Tp __k, _Tpp __phi)
- float std::ellint_1f (float __k, float __phi)

- long double std::ellint_1l (long double __k, long double __phi)
- template<typename _Tp, typename _Tpp>
  __gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::ellint_2 (_Tp __k, _Tpp __phi)
- float std::ellint_2f (float __k, float __phi)
- long double std::ellint_2l (long double __k, long double __phi)
- template<typename _Tp, typename _Tpn, typename _Tpp>
  __gnu_cxx::__promote_3< _Tp, _Tpn, _Tpp >::__type std::ellint_3 (_Tp __k, _Tpn __nu, _Tpp __phi)
- float std::ellint_3f (float __k, float __nu, float __phi)
- long double std::ellint_3l (long double __k, long double __nu, long double __phi)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::expint (_Tp __x)
- float std::expintf (float __x)
- long double std::expintl (long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::hermite (unsigned int __n, _Tp __x)
- float std::hermitef (unsigned int __n, float __x)
- long double std::hermitel (unsigned int __n, long double __x)
- template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp>
  __gnu_cxx::__promote_4< _Tpa, _Tpb, _Tpc, _Tp >::__type __gnu_cxx::hyperg (_Tpa __a, _Tpb __b, _Tpc __c, _Tp __x)
- float __gnu_cxx::hypergf (float __a, float __b, float __c, float __x)
- long double __gnu_cxx::hypergl (long double __a, long double __b, long double __c, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::laguerre (unsigned int __n, _Tp __x)
- float std::laguerref (unsigned int __n, float __x)
- long double std::laguerrel (unsigned int __n, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::legendre (unsigned int __l, _Tp __x)
- float std::legendref (unsigned int __l, float __x)
- long double std::legendrel (unsigned int __l, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::riemann_zeta (_Tp __s)
- float std::riemann_zetaf (float __s)
- long double std::riemann_zetal (long double __s)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::sph_bessel (unsigned int __n, _Tp __x)
- float std::sph_besself (unsigned int __n, float __x)
- long double std::sph_bessell (unsigned int __n, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::sph_legendre (unsigned int __l, unsigned int __m, _Tp __theta)
- float std::sph_legendref (unsigned int __l, unsigned int __m, float __theta)
- long double std::sph_legendrel (unsigned int __l, unsigned int __m, long double __theta)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::sph_neumann (unsigned int __n, _Tp __x)
- float std::sph_neumannf (unsigned int __n, float __x)
- long double std::sph_neumannl (unsigned int __n, long double __x)

6.142.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <cmath>.

6.143 sstream.tcc File Reference

Namespaces

- namespace std

Macros

- #define **_SSTREAM_TCC**

6.143.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <sstream>.

6.144 std_abs.h File Reference

Namespaces

- namespace std

Functions

- constexpr double **std::abs** (double __x)
- constexpr float **std::abs** (float __x)
- long **std::abs** (long __i)
- constexpr long double **std::abs** (long double __x)
- long long **std::abs** (long long __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::abs(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::abs** (simd< _Tp, _Abi > __x)

6.144.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <cmath> or < cstdlib>.

6.145 std_function.h File Reference

Classes

- struct std::__is_location_invariant< _Tp >
- class std::_Function_base
- class std::bad_function_call
- class std::function< _Res(_ArgTypes...)>

Namespaces

- namespace std
- namespace std::__detail

Typedefs

- template<typename _Fn, typename _Op>
  using **std::__function_guide_t**

Enumerations

- enum **_Manager_operation** { **__get_type_info** , **__get_functor_ptr** , **__clone_functor** , **__destroy_functor** }

Functions

- template<typename _Fn, typename _Signature = __function_guide_t<_Fn, decltype(&_Fn::operator())>>
  std::function (_Fn) -> function< _Signature >
- template<typename _Res, typename... _ArgTypes>
  std::function (_Res(*)(_ArgTypes...)) -> function< _Res(_ArgTypes...)>
- template<typename _Res, typename... _Args>
  bool std::operator== (const function< _Res(_Args...)> &__f, nullptr_t) noexcept
- template<typename _Res, typename... _Args>
  void std::swap (function< _Res(_Args...)> &__x, function< _Res(_Args...)> &__y) noexcept

6.145.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <functional>.

6.146 std_mutex.h File Reference

Classes

- struct std::adopt_lock_t
- struct std::defer_lock_t
- class std::lock_guard< _Mutex >
- class std::mutex
- struct std::try_to_lock_t

Namespaces

- namespace std

Variables

- constexpr adopt_lock_t std::adopt_lock
- constexpr defer_lock_t std::defer_lock
- constexpr try_to_lock_t std::try_to_lock

6.146.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <mutex>.

6.147 std_thread.h File Reference

Classes

- struct std::hash< thread::id >
- class std::thread::id
- class std::thread

Namespaces

- namespace std
- namespace std::this_thread

Functions

- thread::id std::this_thread::get_id () noexcept
- void std::this_thread::yield () noexcept

6.147.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <thread>.

6.148 stl_algo.h File Reference

Namespaces

- namespace std

Enumerations

- enum { **_S_chunk_size** }

Functions

- template<typename _ForwardIterator, typename _BinaryPredicate>
  constexpr _ForwardIterator **std::__adjacent_find** (_ForwardIterator __first, _ForwardIterator __last, _Binary↩ Predicate __binary_pred)
- template<typename _RandomAccessIterator, typename _Distance, typename _Compare>
  constexpr void **std::__chunk_insertion_sort** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Distance __chunk_size, _Compare __comp)
- template<typename _ForwardIterator, typename _Tp, typename _Compare>
  constexpr pair< _ForwardIterator, _ForwardIterator > **std::__equal_range** (_ForwardIterator __first, _Forward↩ Iterator __last, const _Tp &__val, _Compare __comp)
- template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _BinaryPredicate>
  constexpr _BidirectionalIterator1 **std::__find_end** (_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, bidirectional_iterator_tag, bidirectional_iterator_tag, _BinaryPredicate __comp)
- template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>
  constexpr _ForwardIterator1 **std::__find_end** (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _Forward↩ Iterator2 __first2, _ForwardIterator2 __last2, forward_iterator_tag, forward_iterator_tag, _BinaryPredicate __↩ comp)
- template<typename _InputIterator, typename _Predicate>
  constexpr _InputIterator std::__find_if_not (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _InputIterator, typename _Predicate, typename _Distance>
  constexpr _InputIterator std::__find_if_not_n (_InputIterator __first, _Distance &__len, _Predicate __pred)
- template<typename _EuclideanRingElement>
  constexpr _EuclideanRingElement std::__gcd (_EuclideanRingElement __m, _EuclideanRingElement __n)
- template<typename _IntType, typename _UniformRandomBitGenerator>
  pair< _IntType, _IntType > std::__gen_two_uniform_ints (_IntType __b0, _IntType __b1, _UniformRandomBit↩ Generator &&__g)

- template<typename _InputIterator1, typename _InputIterator2, typename _Compare>
  constexpr bool **std::__includes** (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Input↩ Iterator2 __last2, _Compare __comp)
- template<typename _BidirectionalIterator, typename _Compare>
  void **std::__inplace_merge** (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Compare __comp)
- template<typename _RandomAccessIterator, typename _Compare>
  void std::__inplace_stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare _↩ _comp)
- template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>
  constexpr bool **std::__is_permutation** (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __pred)
- template<typename _ForwardIterator, typename _Compare>
  constexpr _ForwardIterator **std::__is_sorted_until** (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)
- template<typename _ForwardIterator, typename _Compare>
  constexpr _ForwardIterator **std::__max_element** (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator **std::__merge** (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _BidirectionalIterator, typename _Distance, typename _Pointer, typename _Compare>
  void std::__merge_adaptive (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Pointer __buffer, _Compare __comp)
- template<typename _BidirectionalIterator, typename _Distance, typename _Pointer, typename _Compare>
  void **std::__merge_adaptive_resize** (_BidirectionalIterator __first, _BidirectionalIterator __middle, _↩ BidirectionalIterator __last, _Distance __len1, _Distance __len2, _Pointer __buffer, _Distance __buffer_size, _Compare __comp)
- template<typename _RandomAccessIterator1, typename _RandomAccessIterator2, typename _Distance, typename _Compare>
  void **std::__merge_sort_loop** (_RandomAccessIterator1 __first, _RandomAccessIterator1 __last, _Random↩ AccessIterator2 __result, _Distance __step_size, _Compare __comp)
- template<typename _RandomAccessIterator, typename _Pointer, typename _Compare>
  void **std::__merge_sort_with_buffer** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Pointer __buffer, _Compare __comp)
- template<typename _BidirectionalIterator, typename _Distance, typename _Compare>
  void std::__merge_without_buffer (_BidirectionalIterator __first, _BidirectionalIterator __middle, _Bidirectional↩ Iterator __last, _Distance __len1, _Distance __len2, _Compare __comp)
- template<typename _ForwardIterator, typename _Compare>
  constexpr _ForwardIterator **std::__min_element** (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)
- template<typename _ForwardIterator, typename _Compare>
  constexpr pair< _ForwardIterator, _ForwardIterator > **std::__minmax_element** (_ForwardIterator __first, _↩ ForwardIterator __last, _Compare __comp)
- template<typename _Iterator, typename _Compare>
  constexpr void std::__move_median_to_first (_Iterator __result, _Iterator __a, _Iterator __b, _Iterator __c, _↩ Compare __comp)
- template<typename _InputIterator, typename _OutputIterator, typename _Compare>
  _OutputIterator std::__move_merge (_InputIterator __first1, _InputIterator __last1, _InputIterator __first2, _↩ InputIterator __last2, _OutputIterator __result, _Compare __comp)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  void std::__move_merge_adaptive (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _↩ InputIterator2 __last2, _OutputIterator __result, _Compare __comp)

- template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _BidirectionalIterator3, typename _Compare>
  void std::__move_merge_adaptive_backward (_BidirectionalIterator1 __first1, _BidirectionalIterator1 __last1, _BidirectionalIterator2 __first2, _BidirectionalIterator2 __last2, _BidirectionalIterator3 __result, _Compare __comp)
- template<typename _BidirectionalIterator, typename _Compare>
  constexpr bool **std::__next_permutation** (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)
- template<typename _BidirectionalIterator, typename _Predicate>
  constexpr _BidirectionalIterator std::__partition (_BidirectionalIterator __first, _BidirectionalIterator __last, _Predicate __pred, bidirectional_iterator_tag)
- template<typename _ForwardIterator, typename _Predicate>
  constexpr _ForwardIterator std::__partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, forward_iterator_tag)
- template<typename _BidirectionalIterator, typename _Compare>
  constexpr bool **std::__prev_permutation** (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)
- template<typename _InputIterator, typename _OutputIterator, typename _Predicate>
  constexpr _OutputIterator **std::__remove_copy_if** (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)
- template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _Tp>
  constexpr _OutputIterator **std::__replace_copy_if** (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred, const _Tp &__new_value)
- template<typename _BidirectionalIterator>
  constexpr void std::__reverse (_BidirectionalIterator __first, _BidirectionalIterator __last, bidirectional_iterator_tag)
- template<typename _RandomAccessIterator>
  constexpr void std::__reverse (_RandomAccessIterator __first, _RandomAccessIterator __last, random_access_iterator_tag)
- template<typename _BidirectionalIterator>
  constexpr _BidirectionalIterator std::__rotate (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, bidirectional_iterator_tag)
- template<typename _ForwardIterator>
  constexpr _ForwardIterator std::__rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last, forward_iterator_tag)
- template<typename _RandomAccessIterator>
  constexpr _RandomAccessIterator std::__rotate (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, random_access_iterator_tag)
- template<typename _BidirectionalIterator1, typename _BidirectionalIterator2, typename _Distance>
  _BidirectionalIterator1 std::__rotate_adaptive (_BidirectionalIterator1 __first, _BidirectionalIterator1 __middle, _BidirectionalIterator1 __last, _Distance __len1, _Distance __len2, _BidirectionalIterator2 __buffer, _Distance __buffer_size)
- template<typename _ForwardIterator, typename _OutputIterator, typename _Cat, typename _Size, typename _UniformRandomBitGenerator>
  _OutputIterator std::__sample (_ForwardIterator __first, _ForwardIterator __last, forward_iterator_tag, _OutputIterator __out, _Cat, _Size __n, _UniformRandomBitGenerator &&__g)
- template<typename _InputIterator, typename _RandomAccessIterator, typename _Size, typename _UniformRandomBitGenerator>
  _RandomAccessIterator std::__sample (_InputIterator __first, _InputIterator __last, input_iterator_tag, _RandomAccessIterator __out, random_access_iterator_tag, _Size __n, _UniformRandomBitGenerator &&__g)
- template<typename _ForwardIterator, typename _Integer, typename _UnaryPredicate>
  constexpr _ForwardIterator **std::__search_n** (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, _UnaryPredicate __unary_pred)
- template<typename _ForwardIterator, typename _Integer, typename _UnaryPredicate>
  constexpr _ForwardIterator std::__search_n_aux (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, _UnaryPredicate __unary_pred, std::forward_iterator_tag)
- template<typename _RandomAccessIter, typename _Integer, typename _UnaryPredicate>
  constexpr _RandomAccessIter std::__search_n_aux (_RandomAccessIter __first, _RandomAccessIter __last, _Integer __count, _UnaryPredicate __unary_pred, std::random_access_iterator_tag)

- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator **std::__set_difference** (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator **std::__set_intersection** (_InputIterator1 __first1, _InputIterator1 __last1, _Input↩ Iterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator **std::__set_symmetric_difference** (_InputIterator1 __first1, _InputIterator1 __last1, ↩ _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator **std::__set_union** (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 _↩ _first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _ForwardIterator, typename _Predicate>
  _ForwardIterator **std::__stable_partition** (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)
- template<typename _ForwardIterator, typename _Pointer, typename _Predicate, typename _Distance>
  _ForwardIterator std::__stable_partition_adaptive (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, _Distance __len, _Pointer __buffer, _Distance __buffer_size)
- template<typename _RandomAccessIterator, typename _Compare>
  void **std::__stable_sort** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _RandomAccessIterator, typename _Pointer, typename _Compare>
  void **std::__stable_sort_adaptive** (_RandomAccessIterator __first, _RandomAccessIterator __middle, _↩ RandomAccessIterator __last, _Pointer __buffer, _Compare __comp)
- template<typename _RandomAccessIterator, typename _Pointer, typename _Distance, typename _Compare>
  void **std::__stable_sort_adaptive_resize** (_RandomAccessIterator __first, _RandomAccessIterator __last, _↩ Pointer __buffer, _Distance __buffer_size, _Compare __comp)
- template<typename _ForwardIterator, typename _BinaryPredicate>
  constexpr _ForwardIterator **std::__unique** (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)
- template<typename _ForwardIterator, typename _OutputIterator, typename _BinaryPredicate>
  constexpr _OutputIterator **std::__unique_copy** (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred, forward_iterator_tag)
- template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate>
  constexpr _OutputIterator **std::__unique_copy** (_InputIterator __first, _InputIterator __last, _OutputIterator __↩ result, _BinaryPredicate __binary_pred, input_iterator_tag)
- template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate>
  _ForwardIterator **std::__unique_copy_1** (_InputIterator __first, _InputIterator __last, _ForwardIterator __result, _BinaryPredicate __binary_pred, __true_type)
- template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate>
  constexpr _OutputIterator **std::__unique_copy_1** (_InputIterator __first, _InputIterator __last, _OutputIterator ↩ __result, _BinaryPredicate __binary_pred, __false_type)
- template<typename _ForwardIterator, typename _Tp, typename _Compare>
  constexpr _ForwardIterator **std::__upper_bound** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)
- template<typename _ForwardIterator>
  constexpr _ForwardIterator std::adjacent_find (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _BinaryPredicate>
  constexpr _ForwardIterator std::adjacent_find (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)
- template<typename _InputIterator, typename _Predicate>
  constexpr bool std::all_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _InputIterator, typename _Predicate>
  constexpr bool std::any_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)

- template<typename _ForwardIterator, typename _Tp>
  constexpr bool std::binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)
- template<typename _ForwardIterator, typename _Tp, typename _Compare>
  constexpr bool std::binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)
- template<typename _Tp>
  constexpr const _Tp & std::clamp (const _Tp &__val, const _Tp &__lo, const _Tp &__hi)
- template<typename _Tp, typename _Compare>
  constexpr const _Tp & std::clamp (const _Tp &__val, const _Tp &__lo, const _Tp &__hi, _Compare __comp)
- template<typename _InputIterator, typename _OutputIterator, typename _Predicate>
  constexpr _OutputIterator std::copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)
- template<typename _InputIterator, typename _Size, typename _OutputIterator>
  constexpr _OutputIterator std::copy_n (_InputIterator __first, _Size __n, _OutputIterator __result)
- template<typename _InputIterator, typename _Tp>
  constexpr iterator_traits< _InputIterator >::difference_type std::count (_InputIterator __first, _InputIterator __last, const _Tp &__value)
- template<typename _InputIterator, typename _Predicate>
  constexpr iterator_traits< _InputIterator >::difference_type std::count_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _ForwardIterator, typename _Tp>
  constexpr pair< _ForwardIterator, _ForwardIterator > std::equal_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)
- template<typename _ForwardIterator, typename _Tp, typename _Compare>
  constexpr pair< _ForwardIterator, _ForwardIterator > std::equal_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)
- template<typename _InputIterator, typename _Tp>
  constexpr _InputIterator std::find (_InputIterator __first, _InputIterator __last, const _Tp &__val)
- template<typename _ForwardIterator1, typename _ForwardIterator2>
  constexpr _ForwardIterator1 std::find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)
- template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>
  constexpr _ForwardIterator1 std::find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __comp)
- template<typename _InputIterator, typename _ForwardIterator>
  constexpr _InputIterator std::find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2)
- template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate>
  constexpr _InputIterator std::find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2, _BinaryPredicate __comp)
- template<typename _InputIterator, typename _Predicate>
  constexpr _InputIterator std::find_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _InputIterator, typename _Predicate>
  constexpr _InputIterator std::find_if_not (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _InputIterator, typename _Function>
  constexpr _Function std::for_each (_InputIterator __first, _InputIterator __last, _Function __f)
- template<typename _InputIterator, typename _Size, typename _Function>
  constexpr _InputIterator std::for_each_n (_InputIterator __first, _Size __n, _Function __f)
- template<typename _ForwardIterator, typename _Generator>
  constexpr void std::generate (_ForwardIterator __first, _ForwardIterator __last, _Generator __gen)
- template<typename _OutputIterator, typename _Size, typename _Generator>
  constexpr _OutputIterator std::generate_n (_OutputIterator __first, _Size __n, _Generator __gen)

- template<typename _InputIterator1, typename _InputIterator2>
  constexpr bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)
- template<typename _InputIterator1, typename _InputIterator2, typename _Compare>
  constexpr bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp)
- template<typename _BidirectionalIterator>
  void std::inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last)
- template<typename _BidirectionalIterator, typename _Compare>
  void std::inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Compare __comp)
- template<typename _InputIterator, typename _Predicate>
  constexpr bool std::is_partitioned (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>
  constexpr bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _BinaryPredicate __pred)
- template<typename _ForwardIterator1, typename _ForwardIterator2>
  constexpr bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)
- template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>
  constexpr bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __pred)
- template<typename _ForwardIterator>
  constexpr bool std::is_sorted (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _Compare>
  constexpr bool std::is_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)
- template<typename _ForwardIterator>
  constexpr _ForwardIterator std::is_sorted_until (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _Compare>
  constexpr _ForwardIterator std::is_sorted_until (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)
- template<typename _ForwardIterator, typename _Tp, typename _Compare>
  constexpr _ForwardIterator std::lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)
- template<typename _Tp>
  constexpr _Tp **std::max** (initializer_list< _Tp >)
- template<typename _Tp, typename _Compare>
  constexpr _Tp **std::max** (initializer_list< _Tp >, _Compare)
- template<typename _ForwardIterator>
  constexpr _ForwardIterator std::max_element (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _Compare>
  constexpr _ForwardIterator std::max_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>
  constexpr _OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _Tp>
  constexpr _Tp **std::min** (initializer_list< _Tp >)

- template<typename _Tp, typename _Compare>
  constexpr _Tp **std::min** (initializer_list< _Tp >, _Compare)
- template<typename _ForwardIterator>
  constexpr _ForwardIterator std::min_element (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _Compare>
  constexpr _ForwardIterator std::min_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)
- template<typename _Tp>
  constexpr pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b)
- template<typename _Tp, typename _Compare>
  constexpr pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b, _Compare __comp)
- template<typename _Tp>
  constexpr pair< _Tp, _Tp > **std::minmax** (initializer_list< _Tp >)
- template<typename _Tp, typename _Compare>
  constexpr pair< _Tp, _Tp > **std::minmax** (initializer_list< _Tp >, _Compare)
- template<typename _ForwardIterator>
  constexpr pair< _ForwardIterator, _ForwardIterator > std::minmax_element (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _Compare>
  constexpr pair< _ForwardIterator, _ForwardIterator > std::minmax_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)
- template<typename _BidirectionalIterator>
  constexpr bool std::next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)
- template<typename _BidirectionalIterator, typename _Compare>
  constexpr bool std::next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)
- template<typename _InputIterator, typename _Predicate>
  constexpr bool std::none_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _RandomAccessIterator>
  constexpr void std::nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void std::nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Compare __comp)
- template<typename _RandomAccessIterator>
  constexpr void std::partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void std::partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp)
- template<typename _InputIterator, typename _RandomAccessIterator>
  constexpr _RandomAccessIterator std::partial_sort_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last)
- template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare>
  constexpr _RandomAccessIterator std::partial_sort_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last, _Compare __comp)
- template<typename _ForwardIterator, typename _Predicate>
  constexpr _ForwardIterator std::partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)
- template<typename _InputIterator, typename _OutputIterator1, typename _OutputIterator2, typename _Predicate>
  constexpr pair< _OutputIterator1, _OutputIterator2 > std::partition_copy (_InputIterator __first, _InputIterator __last, _OutputIterator1 __out_true, _OutputIterator2 __out_false, _Predicate __pred)

- template<typename _ForwardIterator, typename _Predicate>
  constexpr _ForwardIterator std::partition_point (_ForwardIterator __first, _ForwardIterator __last, _Predicate _↩
  _pred)
- template<typename _BidirectionalIterator>
  constexpr bool std::prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)
- template<typename _BidirectionalIterator, typename _Compare>
  constexpr bool std::prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __↩
  comp)
- template<typename _RandomAccessIterator>
  void std::random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _RandomNumberGenerator>
  void std::random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomNumber↩
  Generator &&__rand)
- template<typename _ForwardIterator, typename _Tp>
  constexpr _ForwardIterator std::remove (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)
- template<typename _InputIterator, typename _OutputIterator, typename _Tp>
  constexpr _OutputIterator std::remove_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result,
  const _Tp &__value)
- template<typename _InputIterator, typename _OutputIterator, typename _Predicate>
  constexpr _OutputIterator std::remove_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __↩
  result, _Predicate __pred)
- template<typename _ForwardIterator, typename _Predicate>
  constexpr _ForwardIterator std::remove_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)
- template<typename _ForwardIterator, typename _Tp>
  constexpr void std::replace (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__old_value, const _Tp
  &__new_value)
- template<typename _InputIterator, typename _OutputIterator, typename _Tp>
  constexpr _OutputIterator std::replace_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result,
  const _Tp &__old_value, const _Tp &__new_value)
- template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _Tp>
  constexpr _OutputIterator std::replace_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __↩
  result, _Predicate __pred, const _Tp &__new_value)
- template<typename _ForwardIterator, typename _Predicate, typename _Tp>
  constexpr void std::replace_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, const _Tp
  &__new_value)
- template<typename _BidirectionalIterator>
  constexpr void std::reverse (_BidirectionalIterator __first, _BidirectionalIterator __last)
- template<typename _BidirectionalIterator, typename _OutputIterator>
  constexpr _OutputIterator std::reverse_copy (_BidirectionalIterator __first, _BidirectionalIterator __last, _Output↩
  Iterator __result)
- template<typename _ForwardIterator>
  constexpr _ForwardIterator std::rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator _↩
  _last)
- template<typename _ForwardIterator, typename _OutputIterator>
  constexpr _OutputIterator std::rotate_copy (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator
  __last, _OutputIterator __result)
- template<typename _PopulationIterator, typename _SampleIterator, typename _Distance, typename _UniformRandomBitGenerator>
  _SampleIterator std::sample (_PopulationIterator __first, _PopulationIterator __last, _SampleIterator __out, _↩
  Distance __n, _UniformRandomBitGenerator &&__g)
- template<typename _ForwardIterator, typename _Searcher>
  constexpr _ForwardIterator std::search (_ForwardIterator __first, _ForwardIterator __last, const _Searcher &__↩
  searcher)

- template<typename _ForwardIterator1, typename _ForwardIterator2>
  constexpr _ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)
- template<typename _ForwardIterator, typename _Integer, typename _Tp>
  constexpr _ForwardIterator std::search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val)
- template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate>
  constexpr _ForwardIterator std::search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>
  constexpr _OutputIterator std::set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 _↩ _first2, _InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator std::set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 _↩ _first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>
  constexpr _OutputIterator std::set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator std::set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>
  constexpr _OutputIterator std::set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, _↩ InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator std::set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, _↩ InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>
  constexpr _OutputIterator std::set_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator std::set_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator>
  void std::shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _UniformRandomNumber↩ Generator &&__g)
- template<typename _RandomAccessIterator>
  constexpr void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _ForwardIterator, typename _Predicate>
  _ForwardIterator std::stable_partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)
- template<typename _RandomAccessIterator>
  void std::stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _Compare>
  void std::stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation>
  constexpr _OutputIterator std::transform (_InputIterator __first, _InputIterator __last, _OutputIterator __result, ↩ _UnaryOperation __unary_op)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _BinaryOperation>
  constexpr _OutputIterator std::transform (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _OutputIterator __result, _BinaryOperation __binary_op)

- template<typename _ForwardIterator>
  constexpr _ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _BinaryPredicate>
  constexpr _ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)
- template<typename _InputIterator, typename _OutputIterator>
  constexpr _OutputIterator std::unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result)
- template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate>
  constexpr _OutputIterator std::unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred)
- template<typename _ForwardIterator, typename _Tp>
  constexpr _ForwardIterator std::upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)
- template<typename _ForwardIterator, typename _Tp, typename _Compare>
  constexpr _ForwardIterator std::upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)

6.148.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <algorithm>.

6.149 stl_algobase.h File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_ADVANCE**(P, N)
- #define **_GLIBCXX_MOVE3**(_Tp, _Up, _Vp)
- #define **_GLIBCXX_MOVE_BACKWARD3**(_Tp, _Up, _Vp)
- #define **_GLIBCXX_TO_ADDR**(P)

Functions

- template<bool _IsMove, typename _OutIter, typename _InIter>
  constexpr void **std::__assign_one** (_OutIter &__out, _InIter &__in)
- template<bool _IsMove, typename _II, typename _OI>
  constexpr _OI **std::__copy_move_a** (_II __first, _II __last, _OI __result)
- template<bool _IsMove, typename _II, typename _Ite, typename _Seq, typename _Cat>
  constexpr __gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > **std::__copy_move_a** (_II, _II, const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &)
- template<bool _IsMove, typename _IIte, typename _ISeq, typename _ICat, typename _OIte, typename _OSeq, typename _OCat>
  constexpr ::__gnu_debug::_Safe_iterator< _OIte, _OSeq, _OCat > **std::__copy_move_a** (const ::__gnu_debug::_Safe_iterator< _IIte, _ISeq, _ICat > &, const ::__gnu_debug::_Safe_iterator< _IIte, _ISeq, _ICat > &, const ::__gnu_debug::_Safe_iterator< _OIte, _OSeq, _OCat > &)
- template<bool _IsMove, typename _Ite, typename _Seq, typename _Cat, typename _OI>
  constexpr _OI **std::__copy_move_a** (const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &, const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &, _OI)
- template<bool _IsMove, typename _ITp, typename _IRef, typename _IPtr, typename _OTp>
  ::_Deque_iterator< _OTp, _OTp &, _OTp * > **std::__copy_move_a1** (::_Deque_iterator< _ITp, _IRef, _IPtr > __first, ::_Deque_iterator< _ITp, _IRef, _IPtr > __last, ::_Deque_iterator< _OTp, _OTp &, _OTp * > __result)

- template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI>
  _OI **std::__copy_move_a1** (::_Deque_iterator< _Tp, _Ref, _Ptr > __first, ::_Deque_iterator< _Tp, _Ref, _Ptr > __last, _OI __result)
- template<bool _IsMove, typename _II, typename _Tp>
  __gnu_cxx::__enable_if< __is_random_access_iter< _II >::__value,::_Deque_iterator< _Tp, _Tp &, _Tp * > >::__type **std::__copy_move_a1** (_II __first, _II __last, ::_Deque_iterator< _Tp, _Tp &, _Tp * > __result)
- template<bool _IsMove, typename _II, typename _OI>
  constexpr _OI **std::__copy_move_a1** (_II __first, _II __last, _OI __result)
- template<bool _IsMove, typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT, char_traits< _CharT > > >::__type **std::__copy_move_a2** (_CharT *, _CharT *, ostreambuf_iterator< _CharT, char_traits< _CharT > >)
- template<bool _IsMove, typename _InIter, typename _Sent, typename _OutIter>
  constexpr _OutIter **std::__copy_move_a2** (_InIter __first, _Sent __last, _OutIter __result)
- template<bool _IsMove, typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT, char_traits< _CharT > > >::__type **std::__copy_move_a2** (const _CharT *, const _CharT *, ostreambuf_iterator< _CharT, char_traits< _CharT > >)
- template<bool _IsMove, typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value,::_Deque_iterator< _CharT, _CharT &, _CharT * > >::__type **std::__copy_move_a2** (istreambuf_iterator< _CharT, char_traits< _CharT > > __first, istreambuf_iterator< _CharT, char_traits< _CharT > > __last, ::_Deque_iterator< _CharT, _CharT &, _CharT * > __result)
- template<bool _IsMove, typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type **std::__copy_move_a2** (istreambuf_iterator< _CharT, char_traits< _CharT > >, istreambuf_iterator< _CharT, char_traits< _CharT > >, _CharT *)
- template<bool _IsMove, typename _II, typename _OI>
  constexpr _OI **std::__copy_move_backward_a** (_II __first, _II __last, _OI __result)
- template<bool _IsMove, typename _IIte, typename _Seq, typename _Cat>
  constexpr __gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > **std::__copy_move_backward_a** (_II, _II, const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &)
- template<bool _IsMove, typename _IIte, typename _ISeq, typename _ICat, typename _OIte, typename _OSeq, typename _OCat>
  constexpr ::__gnu_debug::_Safe_iterator< _OIte, _OSeq, _OCat > **std::__copy_move_backward_a** (const ::__gnu_debug::_Safe_iterator< _IIte, _ISeq, _ICat > &, const ::__gnu_debug::_Safe_iterator< _IIte, _ISeq, _ICat > &, const ::__gnu_debug::_Safe_iterator< _OIte, _OSeq, _OCat > &)
- template<bool _IsMove, typename _Ite, typename _Seq, typename _Cat, typename _OI>
  constexpr _OI **std::__copy_move_backward_a** (const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &, const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &, _OI)
- template<bool _IsMove, typename _ITp, typename _IRef, typename _IPtr, typename _OTp>
  ::_Deque_iterator< _OTp, _OTp &, _OTp * > **std::__copy_move_backward_a1** (::_Deque_iterator< _ITp, _IRef, _IPtr > __first, ::_Deque_iterator< _ITp, _IRef, _IPtr > __last, ::_Deque_iterator< _OTp, _OTp &, _OTp * > __result)
- template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI>
  _OI **std::__copy_move_backward_a1** (::_Deque_iterator< _Tp, _Ref, _Ptr > __first, ::_Deque_iterator< _Tp, _Ref, _Ptr > __last, _OI __result)
- template<bool _IsMove, typename _BI1, typename _BI2>
  constexpr _BI2 **std::__copy_move_backward_a1** (_BI1 __first, _BI1 __last, _BI2 __result)
- template<bool _IsMove, typename _II, typename _Tp>
  __gnu_cxx::__enable_if< __is_random_access_iter< _II >::__value,::_Deque_iterator< _Tp, _Tp &, _Tp * > >::__type **std::__copy_move_backward_a1** (_II __first, _II __last, ::_Deque_iterator< _Tp, _Tp &, _Tp * > __result)

- template<bool _IsMove, typename _BI1, typename _BI2>
  constexpr _BI2 **std::__copy_move_backward_a2** (_BI1 __first, _BI1 __last, _BI2 __result)
- template<typename _InputIterator, typename _Size, typename _OutputIterator>
  constexpr _OutputIterator **std::__copy_n_a** (_InputIterator __first, _Size __n, _OutputIterator __result, bool)
- template<typename _CharT, typename _Size>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value,::_Deque_iterator< _CharT, _CharT &, _CharT * > >::__type **std::__copy_n_a** (istreambuf_iterator< _CharT, char_traits< _CharT > > __it, _Size __size, ::_Deque_iterator< _CharT, _CharT &, _CharT * > __result, bool __strict)
- template<typename _CharT, typename _Size>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type **std::__copy_n_a** (istreambuf_iterator< _CharT, char_traits< _CharT > >, _Size, _CharT *, bool)
- template<typename _InputIterator, typename _Predicate>
  constexpr iterator_traits< _InputIterator >::difference_type **std::__count_if** (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _II1, typename _II2>
  constexpr bool **std::__equal4** (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)
- template<typename _II1, typename _II2, typename _BinaryPredicate>
  constexpr bool **std::__equal4** (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, _BinaryPredicate __binary_pred)
- template<typename _II1, typename _II2>
  constexpr bool **std::__equal_aux** (_II1 __first1, _II1 __last1, _II2 __first2)
- template<typename _II1, typename _II2, typename _Seq2, typename _Cat2>
  constexpr bool **std::__equal_aux** (_II1, _II1, const ::__gnu_debug::_Safe_iterator< _II2, _Seq2, _Cat2 > &)
- template<typename _II1, typename _Seq1, typename _Cat1, typename _II2>
  constexpr bool **std::__equal_aux** (const ::__gnu_debug::_Safe_iterator< _II1, _Seq1, _Cat1 > &, const ::__gnu_debug::_Safe_iterator< _II1, _Seq1, _Cat1 > &, _II2)
- template<typename _II1, typename _Seq1, typename _Cat1, typename _II2, typename _Seq2, typename _Cat2>
  constexpr bool **std::__equal_aux** (const ::__gnu_debug::_Safe_iterator< _II1, _Seq1, _Cat1 > &, const ::__gnu_debug::_Safe_iterator< _II1, _Seq1, _Cat1 > &, const ::__gnu_debug::_Safe_iterator< _II2, _Seq2, _Cat2 > &)
- template<typename _Tp, typename _Ref, typename _Ptr, typename _II>
  __gnu_cxx::__enable_if< __is_random_access_iter< _II >::__value, bool >::__type **std::__equal_aux1** (::_Deque_iterator< _Tp, _Ref, _Ptr > __first1, ::_Deque_iterator< _Tp, _Ref, _Ptr > __last1, _II __first2)
- template<typename _Tp1, typename _Ref1, typename _Ptr1, typename _Tp2, typename _Ref2, typename _Ptr2>
  bool **std::__equal_aux1** (::_Deque_iterator< _Tp1, _Ref1, _Ptr1 > __first1, ::_Deque_iterator< _Tp1, _Ref1, _Ptr1 > __last1, ::_Deque_iterator< _Tp2, _Ref2, _Ptr2 > __first2)
- template<typename _II, typename _Tp, typename _Ref, typename _Ptr>
  __gnu_cxx::__enable_if< __is_random_access_iter< _II >::__value, bool >::__type **std::__equal_aux1** (_II __first1, _II __last1, ::_Deque_iterator< _Tp, _Ref, _Ptr > __first2)
- template<typename _II1, typename _II2>
  constexpr bool **std::__equal_aux1** (_II1 __first1, _II1 __last1, _II2 __first2)
- template<typename _FIte, typename _Tp>
  constexpr void **std::__fill_a** (_FIte __first, _FIte __last, const _Tp &__value)
- template<typename _Ite, typename _Seq, typename _Cat, typename _Tp>
  constexpr void **std::__fill_a** (const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &, const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &, const _Tp &)
- template<typename _Ite, typename _Cont, typename _Tp>
  constexpr void **std::__fill_a1** (::__gnu_cxx::__normal_iterator< _Ite, _Cont > __first, ::__gnu_cxx::__normal_iterator< _Ite, _Cont > __last, const _Tp &__value)
- constexpr void **std::__fill_a1** (::_Bit_iterator, ::_Bit_iterator, const bool &)
- template<typename _ForwardIterator, typename _Tp>
  constexpr void **std::__fill_a1** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)

- template<typename _Up, typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_byte< _Up >::__value &&(__are_same< _Up, _Tp >::__value||_↩
  _memcpyable_integer< _Tp >::__width), void >::__type **std::__fill_a1** (_Up *__first, _Up *__last, const _Tp &__x)
- template<typename _Tp, typename _VTp>
  void **std::__fill_a1** (const ::_Deque_iterator< _Tp, _Tp &, _Tp * > &__first, const ::_Deque_iterator< _Tp, _Tp &, _Tp * > &__last, const _VTp &__value)
- template<typename _OutputIterator, typename _Size, typename _Tp>
  constexpr _OutputIterator **std::__fill_n_a** (_OutputIterator __first, _Size __n, const _Tp &__value, std::input_iterator_tag)
- template<typename _OutputIterator, typename _Size, typename _Tp>
  constexpr _OutputIterator **std::__fill_n_a** (_OutputIterator __first, _Size __n, const _Tp &__value, std::output_iterator_tag)
- template<typename _OutputIterator, typename _Size, typename _Tp>
  constexpr _OutputIterator **std::__fill_n_a** (_OutputIterator __first, _Size __n, const _Tp &__value, std::random_access_iterator_tag)
- template<typename _Ite, typename _Seq, typename _Cat, typename _Size, typename _Tp>
  constexpr ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > **std::__fill_n_a** (const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &__first, _Size __n, const _Tp &__value, std::input_iterator_tag)
- template<typename _OutputIterator, typename _Size, typename _Tp>
  constexpr _OutputIterator **std::__fill_n_a1** (_OutputIterator __first, _Size __n, const _Tp &__value)
- template<typename _Iterator, typename _Predicate>
  constexpr _Iterator **std::__find_if** (_Iterator __first, _Iterator __last, _Predicate __pred)
- template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>
  constexpr bool **std::__is_permutation** (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _BinaryPredicate __pred)
- template<typename _II1, typename _II2>
  constexpr bool **std::__lexicographical_compare_aux** (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)
- template<typename _II1, typename _Iter2, typename _Seq2, typename _Cat2>
  constexpr bool **std::__lexicographical_compare_aux** (_II1, _II1, const ::__gnu_debug::_Safe_iterator< _Iter2, _Seq2, _Cat2 > &, const ::__gnu_debug::_Safe_iterator< _Iter2, _Seq2, _Cat2 > &)
- template<typename _Iter1, typename _Seq1, typename _Cat1, typename _II2>
  constexpr bool **std::__lexicographical_compare_aux** (const ::__gnu_debug::_Safe_iterator< _Iter1, _Seq1, _↩
  _Cat1 > &, const ::__gnu_debug::_Safe_iterator< _Iter1, _Seq1, _Cat1 > &, _II2, _II2)
- template<typename _Iter1, typename _Seq1, typename _Cat1, typename _Iter2, typename _Seq2, typename _Cat2>
  constexpr bool **std::__lexicographical_compare_aux** (const ::__gnu_debug::_Safe_iterator< _Iter1, _Seq1, _↩
  _Cat1 > &, const ::__gnu_debug::_Safe_iterator< _Iter1, _Seq1, _Cat1 > &, const ::__gnu_debug::_Safe_iterator< _Iter2, _Seq2, _Cat2 > &, const ::__gnu_debug::_Safe_iterator< _Iter2, _Seq2, _Cat2 > &)
- template<typename _Tp1, typename _Ref1, typename _Ptr1, typename _Tp2, typename _Ref2, typename _Ptr2>
  bool **std::__lexicographical_compare_aux1** (::_Deque_iterator< _Tp1, _Ref1, _Ptr1 > __first1, ::_Deque_↩
  iterator< _Tp1, _Ref1, _Ptr1 > __last1, ::_Deque_iterator< _Tp2, _Ref2, _Ptr2 > __first2, ::_Deque_iterator< _Tp2, _Ref2, _Ptr2 > __last2)
- template<typename _Tp1, typename _Ref1, typename _Ptr1, typename _Tp2>
  bool **std::__lexicographical_compare_aux1** (::_Deque_iterator< _Tp1, _Ref1, _Ptr1 > __first1, ::_Deque_↩
  iterator< _Tp1, _Ref1, _Ptr1 > __last1, _Tp2 *__first2, _Tp2 *__last2)
- template<typename _II1, typename _II2>
  constexpr bool **std::__lexicographical_compare_aux1** (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)
- template<typename _Tp1, typename _Tp2, typename _Ref2, typename _Ptr2>
  bool **std::__lexicographical_compare_aux1** (_Tp1 *__first1, _Tp1 *__last1, ::_Deque_iterator< _Tp2, _Ref2, _Ptr2 > __first2, ::_Deque_iterator< _Tp2, _Ref2, _Ptr2 > __last2)
- template<typename _II1, typename _II2, typename _Compare>
  constexpr bool **std::__lexicographical_compare_impl** (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, _↩
  Compare __comp)
- template<typename _Tp>
  constexpr _Tp std::__lg (_Tp __n)

- template<typename _ForwardIterator, typename _Tp, typename _Compare>
  constexpr _ForwardIterator **std::__lower_bound** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)
- template<typename _Tp, typename _Up>
  constexpr int **std::__memcmp** (const _Tp *__first1, const _Up *__first2, size_t __num)
- template<typename _Tp>
  constexpr auto **std::__min_cmp** (_Tp __x, _Tp __y)
- template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>
  constexpr pair< _InputIterator1, _InputIterator2 > **std::__mismatch** (_InputIterator1 __first1, _InputIterator1 ↩ __last1, _InputIterator2 __first2, _BinaryPredicate __binary_pred)
- template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>
  constexpr pair< _InputIterator1, _InputIterator2 > **std::__mismatch** (_InputIterator1 __first1, _InputIterator1 ↩ __last1, _InputIterator2 __first2, _InputIterator2 __last2, _BinaryPredicate __binary_pred)
- template<typename _ForwardIterator, typename _Predicate>
  constexpr _ForwardIterator **std::__remove_if** (_ForwardIterator __first, _ForwardIterator __last, _Predicate __↩ pred)
- template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>
  constexpr _ForwardIterator1 **std::__search** (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _Forward↩ Iterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __predicate)
- constexpr long long **std::__size_to_integer** (double __n)
- constexpr long long **std::__size_to_integer** (float __n)
- constexpr int **std::__size_to_integer** (int __n)
- constexpr long **std::__size_to_integer** (long __n)
- constexpr long long **std::__size_to_integer** (long double __n)
- constexpr long long **std::__size_to_integer** (long long __n)
- constexpr unsigned **std::__size_to_integer** (unsigned __n)
- constexpr unsigned long **std::__size_to_integer** (unsigned long __n)
- constexpr unsigned long long **std::__size_to_integer** (unsigned long long __n)
- template<typename _II, typename _OI>
  constexpr _OI std::copy (_II __first, _II __last, _OI __result)
- template<typename _BI1, typename _BI2>
  constexpr _BI2 std::copy_backward (_BI1 __first, _BI1 __last, _BI2 __result)
- template<typename _II1, typename _II2>
  constexpr bool std::equal (_II1 __first1, _II1 __last1, _II2 __first2)
- template<typename _II1, typename _II2>
  constexpr bool std::equal (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)
- template<typename _IIter1, typename _IIter2, typename _BinaryPredicate>
  constexpr bool std::equal (_IIter1 __first1, _IIter1 __last1, _IIter2 __first2, _BinaryPredicate __binary_pred)
- template<typename _IIter1, typename _IIter2, typename _BinaryPredicate>
  constexpr bool std::equal (_IIter1 __first1, _IIter1 __last1, _IIter2 __first2, _IIter2 __last2, _BinaryPredicate __↩ binary_pred)
- template<typename _ForwardIterator, typename _Tp>
  constexpr void std::fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)
- template<typename _OI, typename _Size, typename _Tp>
  constexpr _OI std::fill_n (_OI __first, _Size __n, const _Tp &__value)
- template<typename _ForwardIterator1, typename _ForwardIterator2>
  constexpr bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 _↩ _first2)
- template<typename _ForwardIterator1, typename _ForwardIterator2>
  constexpr void std::iter_swap (_ForwardIterator1 __a, _ForwardIterator2 __b)
- template<typename _II1, typename _II2>
  constexpr bool std::lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)

- template<typename _II1, typename _II2, typename _Compare>
  constexpr bool std::lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, _Compare __comp)
- template<typename _InputIter1, typename _InputIter2>
  constexpr auto **std::lexicographical_compare_three_way** (_InputIter1 __first1, _InputIter1 __last1, _InputIter2 __first2, _InputIter2 __last2)
- template<typename _InputIter1, typename _InputIter2, typename _Comp>
  constexpr auto std::lexicographical_compare_three_way (_InputIter1 __first1, _InputIter1 __last1, _InputIter2 _↩ _first2, _InputIter2 __last2, _Comp __comp) -> decltype(__comp(∗__first1, ∗__first2))
- template<typename _ForwardIterator, typename _Tp>
  constexpr _ForwardIterator std::lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)
- template<typename _Tp>
  constexpr const _Tp & std::max (const _Tp &__a, const _Tp &__b)
- template<typename _Tp, typename _Compare>
  constexpr const _Tp & std::max (const _Tp &__a, const _Tp &__b, _Compare __comp)
- template<typename _Tp>
  constexpr const _Tp & std::min (const _Tp &__a, const _Tp &__b)
- template<typename _Tp, typename _Compare>
  constexpr const _Tp & std::min (const _Tp &__a, const _Tp &__b, _Compare __comp)
- template<typename _InputIterator1, typename _InputIterator2>
  constexpr pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2)
- template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>
  constexpr pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_pred)
- template<typename _InputIterator1, typename _InputIterator2>
  constexpr pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)
- template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>
  constexpr pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _BinaryPredicate __binary_pred)
- template<typename _II, typename _OI>
  constexpr _OI std::move (_II __first, _II __last, _OI __result)
- template<typename _BI1, typename _BI2>
  constexpr _BI2 std::move_backward (_BI1 __first, _BI1 __last, _BI2 __result)
- template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>
  constexpr _ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __predicate)
- template<typename _ForwardIterator1, typename _ForwardIterator2>
  constexpr _ForwardIterator2 std::swap_ranges (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _↩ ForwardIterator2 __first2)

6.149.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <algorithm>.

6.150 stl_bvector.h File Reference

Classes

- struct std::hash<::vector< bool, _Alloc > >
- class std::vector< bool, _Alloc >

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_ALWAYS_INLINE**

Typedefs

- typedef unsigned long **std::_Bit_type**

Enumerations

- enum { **_S_word_bit** }

Functions

- constexpr void **std::__fill_a1** (::_Bit_iterator, ::_Bit_iterator, const bool &)
- constexpr void **std::__fill_bvector** (_Bit_type *__v, unsigned int __first, unsigned int __last, bool __x) noexcept
- constexpr void **std::__fill_bvector_n** (_Bit_type *, size_t, bool) noexcept

6.150.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <vector>.

6.151 stl_construct.h File Reference

Namespaces

- namespace std

Functions

- template<typename _Tp, typename... _Args>
  constexpr void std::_Construct (_Tp *__p, _Args &&... __args)
- template<typename _T1>
  void **std::_Construct_novalue** (_T1 *__p)
- template<typename _ForwardIterator>
  constexpr void std::_Destroy (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _Tp>
  constexpr void std::_Destroy (_Tp *__pointer)
- template<typename _ForwardIterator, typename _Size>
  constexpr _ForwardIterator std::_Destroy_n (_ForwardIterator __first, _Size __count)
- template<typename _Tp, typename... _Args>
  requires (!is_unbounded_array_v<_Tp>) && requires { ::new((void*)0) _Tp(std::declval<_Args>()...); }
  constexpr _Tp * **std::construct_at** (_Tp *__location, _Args &&... __args) noexcept(noexcept(::new((void *) 0) _Tp(std::declval< _Args >()...)))
- template<typename _ForwardIterator>
  constexpr void **std::destroy** (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _Tp>
  constexpr void **std::destroy_at** (_Tp *__location)
- template<typename _ForwardIterator, typename _Size>
  constexpr _ForwardIterator **std::destroy_n** (_ForwardIterator __first, _Size __count)

6.151.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

6.152 stl_deque.h File Reference

Classes

- class std::_Deque_base< _Tp, _Alloc >
- struct std::_Deque_iterator< _Tp, _Ref, _Ptr >
- class std::deque< _Tp, _Alloc >

Namespaces

- namespace std

Macros

- #define _GLIBCXX_DEQUE_BUF_SIZE

Functions

- constexpr size_t **std::__deque_buf_size** (size_t __size)
- template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  std::deque (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> deque< _ValT, _Allocator >
- template<typename _Tp, typename _Alloc>
  __detail::__synth3way_t< _Tp > std::operator<=> (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)
- template<typename _Tp, typename _Alloc>
  bool std::operator== (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)
- template<typename _Tp, typename _Alloc>
  void std::swap (deque< _Tp, _Alloc > &__x, deque< _Tp, _Alloc > &__y) noexcept(/*conditional */)

6.152.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <deque>.

6.152.2 Macro Definition Documentation

_GLIBCXX_DEQUE_BUF_SIZE

```
#define _GLIBCXX_DEQUE_BUF_SIZE
```

This function controls the size of memory nodes.

Parameters

__size	The size of an element.

Returns

The number (not byte size) of elements per node.

This function started off as a compiler kludge from SGI, but seems to be a useful wrapper around a repeated constant expression. The **512** is tunable (and no other code needs to change), but no investigation has been done since inheriting the SGI code. Touch _GLIBCXX_DEQUE_BUF_SIZE only if you know what you are doing, however: changing it breaks the binary compatibility!!

6.153 stl_function.h File Reference

Classes

- struct std::binary_function< _Arg1, _Arg2, _Result >
- class std::binary_negate< _Predicate >
- class std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >
- class std::const_mem_fun1_t< _Ret, _Tp, _Arg >
- class std::const_mem_fun_ref_t< _Ret, _Tp >
- class std::const_mem_fun_t< _Ret, _Tp >
- struct std::divides< _Tp >
- struct std::divides< void >
- struct std::equal_to< _Tp >
- struct std::greater< _Tp >
- struct std::greater< void >
- struct std::greater_equal< _Tp >
- struct std::greater_equal< void >
- struct std::less< _Tp >
- struct std::less_equal< _Tp >
- struct std::less_equal< void >
- struct std::logical_and< _Tp >
- struct std::logical_and< void >
- struct std::logical_not< _Tp >
- struct std::logical_not< void >
- struct std::logical_or< _Tp >
- struct std::logical_or< void >
- class std::mem_fun1_ref_t< _Ret, _Tp, _Arg >
- class std::mem_fun1_t< _Ret, _Tp, _Arg >
- class std::mem_fun_ref_t< _Ret, _Tp >
- class std::mem_fun_t< _Ret, _Tp >
- struct std::minus< _Tp >
- struct std::minus< void >
- struct std::modulus< _Tp >
- struct std::modulus< void >
- struct std::multiplies< _Tp >
- struct std::multiplies< void >
- struct std::negate< _Tp >
- struct std::negate< void >
- struct std::not_equal_to< _Tp >
- struct std::not_equal_to< void >
- struct std::plus< _Tp >
- class std::pointer_to_binary_function< _Arg1, _Arg2, _Result >
- class std::pointer_to_unary_function< _Arg, _Result >
- struct std::unary_function< _Arg, _Result >
- class std::unary_negate< _Predicate >

Namespaces

- namespace std

Typedefs

- template<typename _Func, typename _SfinaeType>
  using **std::__has_is_transparent_t**

Functions

- template<typename _Ret, typename _Tp>
  const_mem_fun_t< _Ret, _Tp > **std::mem_fun** (_Ret(_Tp::*__f)() const)
- template<typename _Ret, typename _Tp>
  mem_fun_t< _Ret, _Tp > **std::mem_fun** (_Ret(_Tp::*__f)())
- template<typename _Ret, typename _Tp, typename _Arg>
  const_mem_fun1_t< _Ret, _Tp, _Arg > **std::mem_fun** (_Ret(_Tp::*__f)(_Arg) const)
- template<typename _Ret, typename _Tp, typename _Arg>
  mem_fun1_t< _Ret, _Tp, _Arg > **std::mem_fun** (_Ret(_Tp::*__f)(_Arg))
- template<typename _Ret, typename _Tp>
  const_mem_fun_ref_t< _Ret, _Tp > **std::mem_fun_ref** (_Ret(_Tp::*__f)() const)
- template<typename _Ret, typename _Tp>
  mem_fun_ref_t< _Ret, _Tp > **std::mem_fun_ref** (_Ret(_Tp::*__f)())
- template<typename _Ret, typename _Tp, typename _Arg>
  const_mem_fun1_ref_t< _Ret, _Tp, _Arg > **std::mem_fun_ref** (_Ret(_Tp::*__f)(_Arg) const)
- template<typename _Ret, typename _Tp, typename _Arg>
  mem_fun1_ref_t< _Ret, _Tp, _Arg > **std::mem_fun_ref** (_Ret(_Tp::*__f)(_Arg))
- template<typename _Predicate>
  constexpr unary_negate< _Predicate > std::not1 (const _Predicate &__pred)
- template<typename _Predicate>
  constexpr binary_negate< _Predicate > std::not2 (const _Predicate &__pred)
- template<typename _Arg, typename _Result>
  pointer_to_unary_function< _Arg, _Result > std::ptr_fun (_Result(*__x)(_Arg))
- template<typename _Arg1, typename _Arg2, typename _Result>
  pointer_to_binary_function< _Arg1, _Arg2, _Result > std::ptr_fun (_Result(*__x)(_Arg1, _Arg2))

6.153.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <functional>.

6.154 stl_heap.h File Reference

Namespaces

- namespace std

Functions

- template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _Compare>
  constexpr void **std::__adjust_heap** (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __len, _Tp __value, _Compare __comp)
- template<typename _RandomAccessIterator, typename _Compare, typename _Distance>
  constexpr bool **std::__is_heap** (_RandomAccessIterator __first, _Compare __comp, _Distance __n)

- template<typename _RandomAccessIterator, typename _Distance>
  constexpr bool **std::__is_heap** (_RandomAccessIterator __first, _Distance __n)
- template<typename _RandomAccessIterator>
  constexpr bool **std::__is_heap** (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr bool **std::__is_heap** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare _↩
  _comp)
- template<typename _RandomAccessIterator, typename _Distance, typename _Compare>
  constexpr _Distance **std::__is_heap_until** (_RandomAccessIterator __first, _Distance __n, _Compare &__↩
  comp)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void **std::__make_heap** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare
  &__comp)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void **std::__pop_heap** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Random↩
  AccessIterator __result, _Compare &__comp)
- template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _Compare>
  constexpr void **std::__push_heap** (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __top↩
  Index, _Tp __value, _Compare &__comp)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void **std::__sort_heap** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare
  &__comp)
- template<typename _RandomAccessIterator>
  constexpr bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _RandomAccessIterator>
  constexpr _RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator
  __last)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr _RandomAccessIterator std::is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator
  __last, _Compare __comp)
- template<typename _RandomAccessIterator>
  constexpr void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void std::make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare _↩
  _comp)
- template<typename _RandomAccessIterator>
  constexpr void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void std::pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __↩
  comp)
- template<typename _RandomAccessIterator>
  constexpr void std::push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void std::push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __↩
  comp)
- template<typename _RandomAccessIterator>
  constexpr void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void std::sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __↩
  comp)

6.154.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <queue>.

6.155 bits/stl_iterator.h File Reference

Classes

- class std::back_insert_iterator< _Container >
- class std::common_iterator< _It, _Sent >
- class std::counted_iterator< _It >
- class std::front_insert_iterator< _Container >
- class std::insert_iterator< _Container >
- class std::move_iterator< _Iterator >
- class std::move_sentinel< _Sent >
- class std::reverse_iterator< _Iterator >

Namespaces

- namespace __gnu_cxx
- namespace __gnu_debug
- namespace std

Macros

- #define **_GLIBCXX_MAKE_MOVE_IF_NOEXCEPT_ITERATOR**(_Iter)
- #define **_GLIBCXX_MAKE_MOVE_ITERATOR**(_Iter)

Typedefs

- template<typename _InputIterator>
  using **std::__iter_key_t**
- template<typename _InputIterator>
  using **std::__iter_to_alloc_t**
- template<typename _InputIterator>
  using **std::__iter_val_t**

Functions

- template<typename _Iterator, typename _ReturnType = __conditional_t<__move_if_noexcept_cond <typename iterator_traits<_Iterator>::value_type>::value, _Iterator, move_iterator<_Iterator>>>
  constexpr _ReturnType **std::__make_move_if_noexcept_iterator** (_Iterator __i)
- template<typename _Tp, typename _ReturnType = __conditional_t<__move_if_noexcept_cond<_Tp>::value, const _Tp*, move_iterator<_Tp*>>>
  constexpr _ReturnType **std::__make_move_if_noexcept_iterator** (_Tp *__i)
- template<typename _Iterator>
  constexpr reverse_iterator< _Iterator > **std::__make_reverse_iterator** (_Iterator __i)
- template<typename _Container>
  constexpr back_insert_iterator< _Container > std::back_inserter (_Container &__x)
- template<typename _Container>
  constexpr front_insert_iterator< _Container > std::front_inserter (_Container &__x)
- template<typename _Container>
  constexpr insert_iterator< _Container > std::inserter (_Container &__x, std::__detail::__range_iter_t< _Container > __i)

- template<typename _Iterator>
  constexpr move_iterator< _Iterator > **std::make_move_iterator** (_Iterator __i)
- template<typename _Iterator>
  constexpr reverse_iterator< _Iterator > std::make_reverse_iterator (_Iterator __i)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() != __y.base() } -> convertible_to<bool>; }
  constexpr bool **std::operator!=** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _Iterator>
  requires requires { { __x.base() + __n } -> same_as<_Iterator>; }
  constexpr move_iterator< _Iterator > **std::operator+** (typename move_iterator< _Iterator >::difference_type ↩ __n, const move_iterator< _Iterator > &__x)
- template<typename _Iterator>
  constexpr reverse_iterator< _Iterator > **std::operator+** (typename reverse_iterator< _Iterator >::difference_↩ type __n, const reverse_iterator< _Iterator > &__x)
- template<typename _IteratorL, typename _IteratorR>
  constexpr auto **std::operator-** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &↩ __y) -> decltype(__x.base() - __y.base())
- template<typename _IteratorL, typename _IteratorR>
  constexpr auto **std::operator-** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y) -> decltype(__y.base() - __x.base())
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() < __y.base() } -> convertible_to<bool>; }
  constexpr bool **std::operator<** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() > __y.base() } -> convertible_to<bool>; }
  constexpr bool **std::operator>** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __y.base() < __x.base() } -> convertible_to<bool>; }
  constexpr bool **std::operator>=** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() >= __y.base() } -> convertible_to<bool>; }
  constexpr bool **std::operator<=** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<three_way_comparable _Iterator>
  constexpr compare_three_way_result_t< _Iterator > **std::operator<=>** (const move_iterator< _Iterator > &↩ __x, const move_iterator< _Iterator > &__y)
- template<typename _IteratorL, three_way_comparable_with< _IteratorL > _IteratorR>
  constexpr compare_three_way_result_t< _IteratorL, _IteratorR > **std::operator<=>** (const move_iterator< _↩ IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<three_way_comparable _Iterator>
  constexpr compare_three_way_result_t< _Iterator, _Iterator > **std::operator<=>** (const reverse_iterator< _↩ Iterator > &__x, const reverse_iterator< _Iterator > &__y)
- template<typename _IteratorL, three_way_comparable_with< _IteratorL > _IteratorR>
  constexpr compare_three_way_result_t< _IteratorL, _IteratorR > **std::operator<=>** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _Iterator>
  constexpr bool **std::operator==** (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() == __y.base() } -> convertible_to<bool>; }

constexpr bool **std::operator==** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)

- template<typename _Iterator>
  requires requires { { __x.base() == __y.base() } -> convertible_to<bool>; }
  constexpr bool **std::operator==** (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() == __y.base() } -> convertible_to<bool>; }
  constexpr bool std::operator== (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __y.base() < __x.base() } -> convertible_to<bool>; }
  constexpr bool **std::operator>** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() < __y.base() } -> convertible_to<bool>; }
  constexpr bool **std::operator>** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() < __y.base() } -> convertible_to<bool>; }
  constexpr bool **std::operator>=** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() <= __y.base() } -> convertible_to<bool>; }
  constexpr bool **std::operator>=** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)

Variables

- template<typename _Iterator1, typename _Iterator2>
  constexpr bool **std::disable_sized_sentinel_for**< **move_iterator**< **_Iterator1** >, **move_iterator**< **_Iterator2** > >
- template<typename _Iterator1, typename _Iterator2>
  constexpr bool **std::disable_sized_sentinel_for**< **reverse_iterator**< **_Iterator1** >, **reverse_iterator**< **_Iterator2** > >

6.155.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iterator>.
This file implements reverse_iterator, back_insert_iterator, front_insert_iterator, insert_iterator, __normal_iterator, and their supporting functions and overloaded operators.

6.156 debug/stl_iterator.h File Reference

Namespaces

- namespace __gnu_debug

Functions

- template<typename _Iterator>
  constexpr auto **__gnu_debug::__base** (const std::move_iterator< _Iterator > &__it) -> decltype(std::make_move_iterator(__base(__it.base())))

- template<typename _Iterator, typename _Sequence>
  constexpr std::reverse_iterator< _Iterator > **__gnu_debug::__base** (const std::reverse_iterator< _Safe_iterator< _Iterator, _Sequence, std::random_access_iterator_tag > > &__it)
- template<typename _Iterator, typename _Size>
  constexpr bool **__gnu_debug::__can_advance** (const std::move_iterator< _Iterator > &__it, _Size __n)
- template<typename _Iterator, typename _Diff>
  constexpr bool **__gnu_debug::__can_advance** (const std::move_iterator< _Iterator > &__it, const std::pair< _Diff, _Distance_precision > &__dist, int __way)
- template<typename _Iterator, typename _Size>
  constexpr bool **__gnu_debug::__can_advance** (const std::reverse_iterator< _Iterator > &__it, _Size __n)
- template<typename _Iterator, typename _Diff>
  constexpr bool **__gnu_debug::__can_advance** (const std::reverse_iterator< _Iterator > &__it, const std::pair< _Diff, _Distance_precision > &__dist, int __way)
- template<typename _Iterator>
  constexpr _Distance_traits< _Iterator >::__type **__gnu_debug::__get_distance** (const std::move_iterator< _Iterator > &__first, const std::move_iterator< _Iterator > &__last)
- template<typename _Iterator>
  constexpr _Distance_traits< _Iterator >::__type **__gnu_debug::__get_distance** (const std::reverse_iterator< _Iterator > &__first, const std::reverse_iterator< _Iterator > &__last)
- template<typename _Iterator>
  constexpr auto **__gnu_debug::__unsafe** (const std::move_iterator< _Iterator > &__it) -> decltype(std::make_move_iterator(__unsafe(__it.base())))
- template<typename _Iterator>
  constexpr auto **__gnu_debug::__unsafe** (const std::reverse_iterator< _Iterator > &__it) -> decltype(std::__make_reverse_iterator(__unsafe(__it.base())))
- template<typename _Iterator>
  constexpr bool **__gnu_debug::__valid_range** (const std::move_iterator< _Iterator > &__first, const std::move_iterator< _Iterator > &__last, typename _Distance_traits< _Iterator >::__type &__dist)
- template<typename _Iterator>
  constexpr bool **__gnu_debug::__valid_range** (const std::reverse_iterator< _Iterator > &__first, const std::reverse_iterator< _Iterator > &__last, typename _Distance_traits< _Iterator >::__type &__dist)

6.156.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

6.157 stl_iterator_base_funcs.h File Reference

Namespaces

- namespace std
- namespace std::__detail

Functions

- template<typename _BidirectionalIterator, typename _Distance>
  constexpr void **std::__advance** (_BidirectionalIterator &__i, _Distance __n, bidirectional_iterator_tag)
- template<typename _InputIterator, typename _Distance>
  constexpr void **std::__advance** (_InputIterator &__i, _Distance __n, input_iterator_tag)
- template<typename _OutputIterator, typename _Distance>
  void **std::__advance** (_OutputIterator &, _Distance, output_iterator_tag)=delete
- template<typename _RandomAccessIterator, typename _Distance>
  constexpr void **std::__advance** (_RandomAccessIterator &__i, _Distance __n, random_access_iterator_tag)

- template<typename _Tp>
  ptrdiff_t **std::__distance** (::_List_const_iterator< _Tp >, ::_List_const_iterator< _Tp >, input_iterator_tag)
- template<typename _Tp>
  ptrdiff_t **std::__distance** (::_List_iterator< _Tp >, ::_List_iterator< _Tp >, input_iterator_tag)
- template<typename _InputIterator>
  constexpr iterator_traits< _InputIterator >::difference_type **std::__distance** (_InputIterator __first, _InputIterator __last, input_iterator_tag)
- template<typename _OutputIterator>
  void **std::__distance** (_OutputIterator, _OutputIterator, output_iterator_tag)=delete
- template<typename _RandomAccessIterator>
  constexpr iterator_traits< _RandomAccessIterator >::difference_type **std::__distance** (_RandomAccessIterator __first, _RandomAccessIterator __last, random_access_iterator_tag)
- template<typename _InputIterator, typename _Distance>
  constexpr void std::advance (_InputIterator &__i, _Distance __n)
- template<typename _InputIterator>
  constexpr iterator_traits< _InputIterator >::difference_type std::distance (_InputIterator __first, _InputIterator _↩
  _last)
- template<typename _InputIterator>
  constexpr _InputIterator **std::next** (_InputIterator __x, typename iterator_traits< _InputIterator >::difference_type __n=1)
- template<typename _BidirectionalIterator>
  constexpr _BidirectionalIterator **std::prev** (_BidirectionalIterator __x, typename iterator_traits< _Bidirectional↩
  Iterator >::difference_type __n=1)

6.157.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iterator>.
This file contains all of the general iterator-related utility functions, such as distance() and advance().

6.158 stl_iterator_base_types.h File Reference

Classes

- struct std::bidirectional_iterator_tag
- struct std::contiguous_iterator_tag
- struct std::forward_iterator_tag
- struct std::input_iterator_tag
- struct std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >
- struct std::iterator_traits< _Iterator >
- struct std::iterator_traits< _Tp * >
- struct std::output_iterator_tag
- struct std::random_access_iterator_tag

Namespaces

- namespace std

Macros

- #define **_GLIBCXX26_ALGO_DEF_VAL_T**(_Iterator)
- #define **_GLIBCXX26_DEF_VAL_T**(T)

Typedefs

- template<typename _Iter>
  using **std::__iter_category_t**
- template<typename _InIter>
  using **std::_RequireInputIter**

Functions

- template<typename _Iter>
  constexpr iterator_traits< _Iter >::iterator_category std::__iterator_category (const _Iter &)

6.158.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iterator>.
This file contains all of the general iterator-related utility types, such as iterator_traits and struct iterator.

6.159 stl_list.h File Reference

Classes

- class std::_List_base< _Tp, _Alloc >
- struct std::_List_const_iterator< _Tp >
- struct std::_List_iterator< _Tp >
- struct std::_List_node< _Tp >
- struct std::__detail::_List_node_base
- struct std::__detail::_List_node_header
- class std::list< _Tp, _Alloc >

Namespaces

- namespace std
- namespace std::__detail

Macros

- #define **_GLIBCXX_LIST_REMOVE_RETURN_TYPE_TAG**
- #define **_GLIBCXX_USE_ALLOC_PTR_FOR_LIST**

Functions

- template<typename _Tp>
  ptrdiff_t **std::__distance** (::_List_const_iterator< _Tp >, ::_List_const_iterator< _Tp >, input_iterator_tag)
- template<typename _Tp>
  ptrdiff_t **std::__distance** (::_List_iterator< _Tp >, ::_List_iterator< _Tp >, input_iterator_tag)
- template<bool _Const, typename _Ptr>
  ptrdiff_t **std::__distance** (__list::_Iterator< _Const, _Ptr > __first, __list::_Iterator< _Const, _Ptr > __last, input_iterator_tag __tag)
- template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  std::list (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> list< _ValT, _Allocator >
- template<typename _Tp, typename _Alloc>
  __detail::__synth3way_t< _Tp > std::operator<=> (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)

- template<typename _Tp, typename _Alloc>
  bool std::operator== (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)
- template<typename _Tp, typename _Alloc>
  void std::swap (list< _Tp, _Alloc > &__x, list< _Tp, _Alloc > &__y) noexcept(/*conditional */)

6.159.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <list>.

6.160 stl_map.h File Reference

Classes

- class std::map< _Key, _Tp, _Compare, _Alloc >

Namespaces

- namespace std

Functions

- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  std::map (_InputIterator, _InputIterator, _Allocator) -> map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, less< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Compare = less<__iter_key_t<_InputIterator>>, typename _Allocator = allocator<__iter↩ _to_alloc_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  std::map (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> map< __iter_key↩ _t< _InputIterator >, __iter_val_t< _InputIterator >, _Compare, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::map (initializer_list< pair< _Key, _Tp > >, _Allocator) -> map< _Key, _Tp, less< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Compare = less<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  std::map (initializer_list< pair< _Key, _Tp > >, _Compare=_Compare(), _Allocator=_Allocator()) -> map< _↩ Key, _Tp, _Compare, _Allocator >
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  __detail::__synth3way_t< pair< const _Key, _Tp > > std::operator<=> (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  bool std::operator== (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  void std::swap (map< _Key, _Tp, _Compare, _Alloc > &__x, map< _Key, _Tp, _Compare, _Alloc > &__y) noexcept(/*conditional */)

6.160.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <map>.

6.161 stl_multimap.h File Reference

Classes

- class std::multimap< _Key, _Tp, _Compare, _Alloc >

Namespaces

- namespace std

Functions

- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  std::multimap (_InputIterator, _InputIterator, _Allocator) -> multimap< __iter_key_t< _InputIterator >, __iter↩ _val_t< _InputIterator >, less< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Compare = less<__iter_key_t<_InputIterator>>, typename _Allocator = allocator<__iter↩ _to_alloc_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  std::multimap (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> multimap< ↩ __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Compare, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::multimap (initializer_list< pair< _Key, _Tp > >, _Allocator) -> multimap< _Key, _Tp, less< _Key >, _↩ Allocator >
- template<typename _Key, typename _Tp, typename _Compare = less<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  std::multimap (initializer_list< pair< _Key, _Tp > >, _Compare=_Compare(), _Allocator=_Allocator()) -> multimap< _Key, _Tp, _Compare, _Allocator >
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  bool std::operator!= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _↩ Compare, _Alloc > &__y)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  bool std::operator< (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _↩ Compare, _Alloc > &__y)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  bool std::operator<= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _↩ Compare, _Alloc > &__y)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  bool std::operator== (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _↩ Compare, _Alloc > &__y)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  bool std::operator> (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _↩ Compare, _Alloc > &__y)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  bool std::operator>= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _↩ Compare, _Alloc > &__y)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  void std::swap (multimap< _Key, _Tp, _Compare, _Alloc > &__x, multimap< _Key, _Tp, _Compare, _Alloc > &__y) noexcept(/*conditional */)

6.161.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <map>.

6.162 stl_multiset.h File Reference

Classes

- class std::multiset< _Key, _Compare, _Alloc >

Namespaces

- namespace std

Functions

- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  std::multiset (_InputIterator, _InputIterator, _Allocator) -> multiset< typename iterator_traits< _InputIterator >::value_type, less< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Compare = less<typename iterator_traits<_InputIterator>::value_type>, typename _↩ Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  std::multiset (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> multiset< typename iterator_traits< _InputIterator >::value_type, _Compare, _Allocator >
- template<typename _Key, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::multiset (initializer_list< _Key >, _Allocator) -> multiset< _Key, less< _Key >, _Allocator >
- template<typename _Key, typename _Compare = less<_Key>, typename _Allocator = allocator<_Key>, typename = _RequireNot↩ Allocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  std::multiset (initializer_list< _Key >, _Compare=_Compare(), _Allocator=_Allocator()) -> multiset< _Key, _↩ Compare, _Allocator >
- template<typename _Key, typename _Compare, typename _Alloc>
  bool std::operator!= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)
- template<typename _Key, typename _Compare, typename _Alloc>
  bool std::operator< (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)
- template<typename _Key, typename _Compare, typename _Alloc>
  bool std::operator<= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)
- template<typename _Key, typename _Compare, typename _Alloc>
  bool std::operator== (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)
- template<typename _Key, typename _Compare, typename _Alloc>
  bool std::operator> (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)
- template<typename _Key, typename _Compare, typename _Alloc>
  bool std::operator>= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)
- template<typename _Key, typename _Compare, typename _Alloc>
  void std::swap (multiset< _Key, _Compare, _Alloc > &__x, multiset< _Key, _Compare, _Alloc > &__↩ y) noexcept(/*conditional */)

6.162.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <set>.

6.163 stl_numeric.h File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_MOVE_IF_20**(_E)

Functions

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

6.163.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <numeric>.

6.164 stl_pair.h File Reference

Classes

- struct std::pair< _T1, _T2 >
- struct std::piecewise_construct_t
- struct std::tuple_element< 0, pair< _Tp1, _Tp2 > >
- struct std::tuple_element< 1, pair< _Tp1, _Tp2 > >
- struct std::tuple_size< pair< _Tp1, _Tp2 > >

Namespaces

- namespace std

Functions

- template<typename _T1, typename _T2>
  constexpr pair< typename __decay_and_strip< _T1 >::__type, typename __decay_and_strip< _T2 >::__type > std::make_pair (_T1 &&__x, _T2 &&__y)

- template<typename _Tp, typename _Up>
  constexpr const _Tp && std::get (const pair< _Tp, _Up > &&__p) noexcept
- template<typename _Tp, typename _Up>
  constexpr const _Tp & std::get (const pair< _Tp, _Up > &__p) noexcept
- template<size_t _Int, class _Tp1, class _Tp2>
  constexpr const tuple_element< _Int, pair< _Tp1, _Tp2 > >::type && std::get (const pair< _Tp1, _Tp2 > &&__in) noexcept
- template<size_t _Int, class _Tp1, class _Tp2>
  constexpr const tuple_element< _Int, pair< _Tp1, _Tp2 > >::type & std::get (const pair< _Tp1, _Tp2 > &__in) noexcept
- template<typename _Tp, typename _Up>
  constexpr const _Tp && std::get (const pair< _Up, _Tp > &&__p) noexcept
- template<typename _Tp, typename _Up>
  constexpr const _Tp & std::get (const pair< _Up, _Tp > &__p) noexcept
- template<typename _Tp, typename _Up>
  constexpr _Tp && std::get (pair< _Tp, _Up > &&__p) noexcept
- template<typename _Tp, typename _Up>
  constexpr _Tp & std::get (pair< _Tp, _Up > &__p) noexcept
- template<size_t _Int, class _Tp1, class _Tp2>
  constexpr tuple_element< _Int, pair< _Tp1, _Tp2 > >::type && std::get (pair< _Tp1, _Tp2 > &&__in) noexcept
- template<size_t _Int, class _Tp1, class _Tp2>
  constexpr tuple_element< _Int, pair< _Tp1, _Tp2 > >::type & std::get (pair< _Tp1, _Tp2 > &__in) noexcept
- template<typename _Tp, typename _Up>
  constexpr _Tp && std::get (pair< _Up, _Tp > &&__p) noexcept
- template<typename _Tp, typename _Up>
  constexpr _Tp & std::get (pair< _Up, _Tp > &__p) noexcept

Variables

- template<typename _Tp>
  constexpr bool **std::__is_pair**
- template<typename _Tp, typename _Up>
  constexpr bool **std::__is_pair**< **pair**< **_Tp, _Up** > >
- constexpr piecewise_construct_t std::piecewise_construct
- template<typename _Tp1, typename _Tp2>
  constexpr size_t **std::tuple_size_v**< **const pair**< **_Tp1, _Tp2** > >
- template<typename _Tp1, typename _Tp2>
  constexpr size_t **std::tuple_size_v**< **pair**< **_Tp1, _Tp2** > >

6.164.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <utility>.

6.165 stl_queue.h File Reference

Classes

- class std::priority_queue< _Tp, _Sequence, _Compare >
- class std::queue< _Tp, _Sequence >

Namespaces

- namespace std

Functions

- template<typename _Tp, typename _Seq>
  bool std::operator!= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)
- template<typename _Tp, typename _Seq>
  bool std::operator< (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)
- template<typename _Tp, typename _Seq>
  bool std::operator<= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)
- template<typename _Tp, three_way_comparable _Seq>
  compare_three_way_result_t< _Seq > **std::operator<=>** (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)
- template<typename _Tp, typename _Seq>
  bool std::operator== (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)
- template<typename _Tp, typename _Seq>
  bool std::operator> (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)
- template<typename _Tp, typename _Seq>
  bool std::operator>= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)
- template<typename _Compare, typename _Container, typename = _RequireNotAllocator<_Compare>, typename = _RequireNotAllocator<_Container>>
  std::priority_queue (_Compare, _Container) -> priority_queue< typename _Container::value_type, _Container, _Compare >
- template<typename _Compare, typename _Container, typename _Allocator, typename = _RequireNotAllocator<_Compare>, typename = _RequireNotAllocator<_Container>>
  std::priority_queue (_Compare, _Container, _Allocator) -> priority_queue< typename _Container::value_type, _Container, _Compare >
- template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Compare = less<_ValT>, typename _Container = vector<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireNotAllocator<_Container>>
  std::priority_queue (_InputIterator, _InputIterator, _Compare=_Compare(), _Container=_Container()) -> priority_queue< _ValT, _Container, _Compare >
- template<typename _Container, typename = _RequireNotAllocator<_Container>>
  std::queue (_Container) -> queue< typename _Container::value_type, _Container >
- template<typename _Container, typename _Allocator, typename = _RequireNotAllocator<_Container>>
  std::queue (_Container, _Allocator) -> queue< typename _Container::value_type, _Container >
- template<typename _Tp, typename _Sequence, typename _Compare>
  enable_if< __and_< __is_swappable< _Sequence >, __is_swappable< _Compare > >::value >::type **std::swap** (priority_queue< _Tp, _Sequence, _Compare > &__x, priority_queue< _Tp, _Sequence, _Compare > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename _Tp, typename _Seq>
  enable_if< __is_swappable< _Seq >::value >::type **std::swap** (queue< _Tp, _Seq > &__x, queue< _Tp, _Seq > &__y) noexcept(noexcept(__x.swap(__y)))

6.165.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <queue>.

6.166 stl_raw_storage_iter.h File Reference

Classes

- class std::raw_storage_iterator< _OutputIterator, _Tp >

Namespaces

- namespace std

6.166.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

6.167 stl_relops.h File Reference

Namespaces

- namespace std
- namespace std::rel_ops

Functions

- template<class _Tp>
  bool std::rel_ops::operator!= (const _Tp &__x, const _Tp &__y)
- template<class _Tp>
  bool std::rel_ops::operator<= (const _Tp &__x, const _Tp &__y)
- template<class _Tp>
  bool std::rel_ops::operator> (const _Tp &__x, const _Tp &__y)
- template<class _Tp>
  bool std::rel_ops::operator>= (const _Tp &__x, const _Tp &__y)

6.167.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <utility>.
This file is only included by <utility>, which is required by the standard to define namespace rel_ops and its contents.

6.168 stl_set.h File Reference

Classes

- class std::set< _Key, _Compare, _Alloc >

Namespaces

- namespace std

Functions

- template<typename _Key, typename _Compare, typename _Alloc>
  bool std::operator!= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)
- template<typename _Key, typename _Compare, typename _Alloc>
  bool std::operator< (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)

- template<typename _Key, typename _Compare, typename _Alloc>
  bool std::operator<= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)
- template<typename _Key, typename _Compare, typename _Alloc>
  bool std::operator== (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)
- template<typename _Key, typename _Compare, typename _Alloc>
  bool std::operator> (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)
- template<typename _Key, typename _Compare, typename _Alloc>
  bool std::operator>= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  std::set (_InputIterator, _InputIterator, _Allocator) -> set< typename iterator_traits< _InputIterator >::value_↩ type, less< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Compare = less<typename iterator_traits<_InputIterator>::value_type>, typename _↩ Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  std::set (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> set< typename iterator_traits< _InputIterator >::value_type, _Compare, _Allocator >
- template<typename _Key, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::set (initializer_list< _Key >, _Allocator) -> set< _Key, less< _Key >, _Allocator >
- template<typename _Key, typename _Compare = less<_Key>, typename _Allocator = allocator<_Key>, typename = _RequireNot↩ Allocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  std::set (initializer_list< _Key >, _Compare=_Compare(), _Allocator=_Allocator()) -> set< _Key, _Compare, _Allocator >
- template<typename _Key, typename _Compare, typename _Alloc>
  void std::swap (set< _Key, _Compare, _Alloc > &__x, set< _Key, _Compare, _Alloc > &__y) noexcept(/*conditional */)

6.168.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <set>.

6.169 stl_stack.h File Reference

Classes

- class std::stack< _Tp, _Sequence >

Namespaces

- namespace std

Functions

- template<typename _Tp, typename _Seq>
  bool std::operator!= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)
- template<typename _Tp, typename _Seq>
  bool std::operator< (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)
- template<typename _Tp, typename _Seq>
  bool std::operator<= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)
- template<typename _Tp, three_way_comparable _Seq>
  compare_three_way_result_t< _Seq > **std::operator<=>** (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)

- template<typename _Tp, typename _Seq>
  bool std::operator== (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)
- template<typename _Tp, typename _Seq>
  bool std::operator> (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)
- template<typename _Tp, typename _Seq>
  bool std::operator>= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)
- template<typename _Container, typename = _RequireNotAllocator<_Container>>
  std::stack (_Container) -> stack< typename _Container::value_type, _Container >
- template<typename _Container, typename _Allocator, typename = _RequireNotAllocator<_Container>>
  std::stack (_Container, _Allocator) -> stack< typename _Container::value_type, _Container >
- template<typename _Tp, typename _Seq>
  enable_if< __is_swappable< _Seq >::value >::type **std::swap** (stack< _Tp, _Seq > &__x, stack< _Tp, _Seq > &__y) noexcept(noexcept(__x.swap(__y)))

6.169.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <stack>.

6.170 stl_tempbuf.h File Reference

Classes

- class std::_Temporary_buffer< _ForwardIterator, _Tp >

Namespaces

- namespace std
- namespace std::__detail

Macros

- #define **_GLIBCXX_OPERATOR_DELETE**
- #define **_GLIBCXX_OPERATOR_NEW**
- #define **_GLIBCXX_SIZED_DEALLOC**(T, p, n)

Functions

- template<typename _Tp>
  _Tp * **std::__detail::__get_temporary_buffer** (ptrdiff_t __len) noexcept
- template<typename _Tp>
  void **std::__detail::__return_temporary_buffer** (_Tp *__p, size_t __len)
- template<typename _Tp, typename _ForwardIterator>
  void **std::__uninitialized_construct_buf** (_Tp *__first, _Tp *__last, _ForwardIterator __seed)
- template<typename _Tp>
  pair< _Tp *, ptrdiff_t > std::get_temporary_buffer (ptrdiff_t __len) noexcept
- template<typename _Tp>
  void std::return_temporary_buffer (_Tp *__p)

6.170.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

6.171 stl_tree.h File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_USE_ALLOC_PTR_FOR_RB_TREE**

Enumerations

- enum **_Rb_tree_color** { **_S_red** , **_S_black** }

Functions

- unsigned int **std::_Rb_tree_black_count** (const _Rb_tree_node_base *__node, const _Rb_tree_node_base *__root) throw ()
- _Rb_tree_node_base * **std::_Rb_tree_decrement** (_Rb_tree_node_base *__x) throw ()
- _Rb_tree_node_base * **std::_Rb_tree_increment** (_Rb_tree_node_base *__x) throw ()
- void **std::_Rb_tree_insert_and_rebalance** (const bool __insert_left, _Rb_tree_node_base *__x, _Rb_tree_node_base *__p, _Rb_tree_node_base &__header) throw ()
- _Rb_tree_node_base * **std::_Rb_tree_rebalance_for_erase** (_Rb_tree_node_base *const __z, _Rb_tree_node_base &__header) throw ()
- template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc>
  void **std::swap** (_Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)

6.171.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <map> or <set>.

6.172 stl_uninitialized.h File Reference

Namespaces

- namespace std

Functions

- template<typename _InputIterator, typename _ForwardIterator>
  _ForwardIterator std::uninitialized_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __result)
- template<typename _InputIterator, typename _Size, typename _ForwardIterator>
  _ForwardIterator std::uninitialized_copy_n (_InputIterator __first, _Size __n, _ForwardIterator __result)
- template<typename _ForwardIterator>
  void std::uninitialized_default_construct (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _Size>
  _ForwardIterator std::uninitialized_default_construct_n (_ForwardIterator __first, _Size __count)
- template<typename _ForwardIterator, typename _Tp>
  void std::uninitialized_fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__x)
- template<typename _ForwardIterator, typename _Size, typename _Tp>
  _ForwardIterator std::uninitialized_fill_n (_ForwardIterator __first, _Size __n, const _Tp &__x)
- template<typename _InputIterator, typename _ForwardIterator>
  _ForwardIterator std::uninitialized_move (_InputIterator __first, _InputIterator __last, _ForwardIterator __result)

- template<typename _InputIterator, typename _Size, typename _ForwardIterator>
  pair< _InputIterator, _ForwardIterator > std::uninitialized_move_n (_InputIterator __first, _Size __count, _↩
  ForwardIterator __result)
- template<typename _ForwardIterator>
  void std::uninitialized_value_construct (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _Size>
  _ForwardIterator std::uninitialized_value_construct_n (_ForwardIterator __first, _Size __count)

6.172.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

6.173 stl_vector.h File Reference

Classes

- struct std::_Vector_base< _Tp, _Alloc >
- class std::vector< _Tp, _Alloc >

Namespaces

- namespace std
- namespace std::__detail

Macros

- #define **_GLIBCXX_ASAN_ANNOTATE_BEFORE_DEALLOC**
- #define **_GLIBCXX_ASAN_ANNOTATE_GREW**(n)
- #define **_GLIBCXX_ASAN_ANNOTATE_GROW**(n)
- #define **_GLIBCXX_ASAN_ANNOTATE_REINIT**
- #define **_GLIBCXX_ASAN_ANNOTATE_SHRINK**(n)

Functions

- template<typename _Tp, typename _Alloc>
  constexpr __detail::__synth3way_t< _Tp > std::operator<=> (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)
- template<typename _Tp, typename _Alloc>
  constexpr bool std::operator== (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)
- template<typename _Tp, typename _Alloc>
  constexpr void std::swap (vector< _Tp, _Alloc > &__x, vector< _Tp, _Alloc > &__y) noexcept(/*conditional */)
- template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  std::vector (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> vector< _ValT, _Allocator >

6.173.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <vector>.

6.174 stream_iterator.h File Reference

Classes

- class std::istream_iterator< _Tp, _CharT, _Traits, _Dist >
- class std::ostream_iterator< _Tp, _CharT, _Traits >

Namespaces

- namespace std

6.174.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iterator>.

6.175 streambuf.tcc File Reference

Namespaces

- namespace std

Macros

- #define **_STREAMBUF_TCC**

Functions

- template<typename _CharT, typename _Traits>
  streamsize **std::__copy_streambufs** (basic_streambuf< _CharT, _Traits > *__sbin, basic_streambuf< _CharT, _Traits > *__sbout)
- template<typename _CharT, typename _Traits>
  streamsize **std::__copy_streambufs_eof** (basic_streambuf< _CharT, _Traits > *__sbin, basic_streambuf< _CharT, _Traits > *__sbout, bool &__ineof)

6.175.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <streambuf>.

6.176 streambuf_iterator.h File Reference

Classes

- class std::istreambuf_iterator< _CharT, _Traits >
- class std::ostreambuf_iterator< _CharT, _Traits >

Namespaces

- namespace std

Functions

- template<bool _IsMove, typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type **std::__copy_move_a2** (_CharT *__first, _CharT *__last, ostreambuf_iterator< _CharT > __result)

- template<bool _IsMove, typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type **std::__copy_move_a2** (const _CharT *__first, const _CharT *__last, ostreambuf_iterator< _CharT > __result)
- template<bool _IsMove, typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type **std::__copy_move_a2** (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, _CharT *__result)
- template<typename _CharT, typename _Size>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type **std::__copy_n_a** (istreambuf_iterator< _CharT > __it, _Size __n, _CharT *__result, bool __strict)
- template<typename _CharT, typename _Distance>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, void >::__type **std::advance** (istreambuf_iterator< _CharT > &__i, _Distance __n)
- template<typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type **std::copy** (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, ostreambuf_iterator< _CharT > __result)
- template<typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, istreambuf_iterator< _CharT > >::__type **std::find** (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, const _CharT &__val)
- template<typename _CharT, typename _Traits>
  bool **std::operator==** (const istreambuf_iterator< _CharT, _Traits > &__a, const istreambuf_iterator< _CharT, _Traits > &__b)

6.176.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iterator>.

6.177 bits/string_view.tcc File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_STRING_VIEW_TCC**

6.177.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <string_view>.

6.178 experimental/bits/string_view.tcc File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **_GLIBCXX_EXPERIMENTAL_STRING_VIEW_TCC**

6.178.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <experimental/string_view>.

6.179 stringfwd.h File Reference

Namespaces

- namespace std

Typedefs

- typedef basic_string< char > std::string
- typedef basic_string< char16_t > std::u16string
- typedef basic_string< char32_t > std::u32string
- typedef basic_string< wchar_t > std::wstring

6.179.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <string>.

6.180 text_encoding-data.h File Reference

Macros

- #define **_GLIBCXX_TEXT_ENCODING_UTF8_OFFSET**

6.180.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <text_encoding>.

6.181 this_thread_sleep.h File Reference

Namespaces

- namespace std
- namespace std::this_thread

Functions

- template<typename _Rep, typename _Period>
  void std::this_thread::sleep_for (const chrono::duration< _Rep, _Period > &__rtime)
- template<typename _Clock, typename _Duration>
  void std::this_thread::sleep_until (const chrono::time_point< _Clock, _Duration > &__atime)

6.181.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <thread>.

6.182 unicode-data.h File Reference

Enumerations

- enum class **_Gcb_property** {
  _Gcb_Other , **_Gcb_Control** , **_Gcb_LF** , **_Gcb_CR** ,
  _Gcb_Extend , **_Gcb_Prepend** , **_Gcb_SpacingMark** , **_Gcb_L** ,
  _Gcb_V , **_Gcb_T** , **_Gcb_ZWJ** , **_Gcb_LV** ,
  _Gcb_LVT , **_Gcb_Regional_Indicator** }
- enum class **_InCB** { **_Consonant** , **_Extend** }

Variables

- constexpr uint32_t **__escape_edges** [ ]
- constexpr uint32_t **__gcb_edges** [ ]
- constexpr int **__gcb_shift_bits**
- constexpr uint32_t **__incb_edges** [ ]
- constexpr char32_t **__incb_linkers** [ ]
- constexpr char32_t **__width_edges** [ ]
- constexpr char32_t **__xpicto_edges** [ ]

6.182.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <format>.

6.183 unicode.h File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_GET_UNICODE_DATA**

Typedefs

- template<typename _View>
  using **std::__unicode::_Utf16_view**
- template<typename _View>
  using **std::__unicode::_Utf32_view**
- template<typename _View>
  using **std::__unicode::_Utf8_view**

Enumerations

- enum class **_Gcb_property** {
  _Gcb_Other , **_Gcb_Control** , **_Gcb_LF** , **_Gcb_CR** ,
  _Gcb_Extend , **_Gcb_Prepend** , **_Gcb_SpacingMark** , **_Gcb_L** ,
  _Gcb_V , **_Gcb_T** , **_Gcb_ZWJ** , **_Gcb_LV** ,
  _Gcb_LVT , **_Gcb_Regional_Indicator** }
- enum class **_InCB** { **_Consonant** , **_Extend** }

Functions

- constexpr bool **std::__unicode::__charset_alias_match** (string_view __a, string_view __b)
- template<typename _CharT>
  constexpr size_t **std::__unicode::__field_width** (basic_string_view< _CharT > __s)
- constexpr int **std::__unicode::__field_width** (char32_t __c) noexcept
- constexpr _Gcb_property **std::__unicode::__grapheme_cluster_break_property** (char32_t __c) noexcept
- constexpr _InCB **std::__unicode::__incb_property** (char32_t __c) noexcept
- constexpr bool **std::__unicode::__is_extended_pictographic** (char32_t __c)
- constexpr bool **std::__unicode::__is_incb_linker** (char32_t __c) noexcept
- constexpr bool **std::__unicode::__is_scalar_value** (char32_t __c)
- template<typename _CharT>
  constexpr bool **std::__unicode::__is_single_code_unit** (char32_t __c)
- consteval bool **std::__unicode::__literal_encoding_is_extended_ascii** ()
- template<typename _CharT>
  consteval bool **std::__unicode::__literal_encoding_is_unicode** ()
- consteval bool **std::__unicode::__literal_encoding_is_utf8** ()
- constexpr bool **std::__unicode::__should_escape_category** (char32_t __c) noexcept
- template<typename _CharT>
  constexpr size_t **std::__unicode::__truncate** (basic_string_view< _CharT > &__s, size_t __max)

Variables

- constexpr uint32_t **std::__unicode::__escape_edges** [ ]
- constexpr uint32_t **std::__unicode::__gcb_edges** [ ]
- constexpr int **std::__unicode::__gcb_shift_bits**
- constexpr uint32_t **std::__unicode::__incb_edges** [ ]
- constexpr char32_t **std::__unicode::__incb_linkers** [ ]
- constexpr char32_t **std::__unicode::__width_edges** [ ]
- constexpr char32_t **std::__unicode::__xpicto_edges** [ ]
- template<typename _Range>
  constexpr bool **std::ranges::enable_borrowed_range**< **std::__unicode::_Grapheme_cluster_view**< _↩ **Range** > >
- template<typename _To, typename _Range>
  constexpr bool **std::ranges::enable_borrowed_range**< **std::__unicode::_Utf_view**< **_To, _Range** > >

6.183.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <format>.

6.184 uniform_int_dist.h File Reference

Classes

- struct std::uniform_int_distribution< _IntType >::param_type
- class std::uniform_int_distribution< _IntType >

Namespaces

- namespace std

6.184.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <random>.

6.185 unique_lock.h File Reference

Classes

- class std::unique_lock< _Mutex >

Namespaces

- namespace std

6.185.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <mutex>.

6.186 unique_ptr.h File Reference

Classes

- struct std::default_delete< _Tp >
- struct std::default_delete< _Tp[ ]>
- struct std::hash< unique_ptr< _Tp, _Dp > >
- class std::unique_ptr< _Tp, _Dp >
- class std::unique_ptr< _Tp[ ], _Dp >

Namespaces

- namespace std
- namespace std::__detail

Functions

- template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
  constexpr bool std::operator< (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp, typename _Dp>
  constexpr bool std::operator< (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<typename _Tp, typename _Dp>
  constexpr bool std::operator< (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)
- template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
  constexpr bool std::operator<= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp, typename _Dp>
  constexpr bool std::operator<= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<typename _Tp, typename _Dp>
  constexpr bool std::operator<= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)
- template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
  requires three_way_comparable_with<typename unique_ptr<_Tp, _Dp>::pointer, typename unique_ptr<_Up, _Ep>::pointer>
  constexpr compare_three_way_result_t< typename unique_ptr< _Tp, _Dp >::pointer, typename unique_ptr< _Up, _Ep >::pointer > **std::operator<=>** (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)

- template<typename _Tp, typename _Dp>
  requires three_way_comparable<typename unique_ptr<_Tp, _Dp>::pointer>
  constexpr compare_three_way_result_t< typename unique_ptr< _Tp, _Dp >::pointer > **std::operator<=>** (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
  constexpr bool std::operator== (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp, typename _Dp>
  constexpr bool std::operator== (const unique_ptr< _Tp, _Dp > &__x, nullptr_t) noexcept
- template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
  constexpr bool std::operator> (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp, typename _Dp>
  constexpr bool std::operator> (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<typename _Tp, typename _Dp>
  constexpr bool std::operator> (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)
- template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
  constexpr bool std::operator>= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp, typename _Dp>
  constexpr bool std::operator>= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<typename _Tp, typename _Dp>
  bool std::operator>= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)
- template<typename _Tp, typename _Dp>
  enable_if<!__is_swappable< _Dp >::value >::type **std::swap** (unique_ptr< _Tp, _Dp > &, unique_ptr< _Tp, _Dp > &)=delete

6.186.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

6.187 unordered_map.h File Reference

Classes

- class std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >
- class std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >

Namespaces

- namespace std

Typedefs

- template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >, typename _Tr = __umap_traits<__cache_default<_Key, _Hash>::value>>
  using **std::__umap_hashtable**
- template<bool _Cache>
  using std::__umap_traits
- template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >, typename _Tr = __ummap_traits<__cache_default<_Key, _Hash>::value>>
  using **std::__ummap_hashtable**
- template<bool _Cache>
  using std::__ummap_traits

Functions

- template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc>
  bool **std::operator==** (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc>
  bool **std::operator==** (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc>
  void **std::swap** (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))
- template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc>
  void **std::swap** (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  std::unordered_map (_InputIterator, _InputIterator, _Allocator) -> unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator > >, equal_to< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  std::unordered_map (_InputIterator, _InputIterator, typename unordered_map< int, int >::size_type, _Allocator) -> unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator > >, equal_to< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  std::unordered_map (_InputIterator, _InputIterator, typename unordered_map< int, int >::size_type, _Hash, _Allocator) -> unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Hash, equal_to< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Hash = hash<__iter_key_t<_InputIterator>>, typename _Pred = equal_to<__iter_key_t<_InputIterator>>, typename _Allocator = allocator<__iter_to_alloc_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  std::unordered_map (_InputIterator, _InputIterator, typename unordered_map< int, int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Hash, _Pred, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::unordered_map (initializer_list< pair< _Key, _Tp > >, _Allocator) -> unordered_map< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::unordered_map (initializer_list< pair< _Key, _Tp > >, typename unordered_map< int, int >::size_type, _Allocator) -> unordered_map< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  std::unordered_map (initializer_list< pair< _Key, _Tp > >, typename unordered_map< int, int >::size_type, _Hash, _Allocator) -> unordered_map< _Key, _Tp, _Hash, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  std::unordered_map (initializer_list< pair< _Key, _Tp > >, typename unordered_map< int, int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_map< _Key, _Tp, _Hash, _Pred, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>

std::unordered_multimap (_InputIterator, _InputIterator, _Allocator) -> unordered_multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator > >, equal_to< __iter_key_t< _InputIterator > >, _Allocator >

- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  std::unordered_multimap (_InputIterator, _InputIterator, unordered_multimap< int, int >::size_type, _Allocator) -> unordered_multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator > >, equal_to< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  std::unordered_multimap (_InputIterator, _InputIterator, unordered_multimap< int, int >::size_type, _Hash, _Allocator) -> unordered_multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Hash, equal_to< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Hash = hash<__iter_key_t<_InputIterator>>, typename _Pred = equal_to<__iter_key_t<_InputIterator>>, typename _Allocator = allocator<__iter_to_alloc_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  std::unordered_multimap (_InputIterator, _InputIterator, unordered_multimap< int, int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Hash, _Pred, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::unordered_multimap (initializer_list< pair< _Key, _Tp > >, _Allocator) -> unordered_multimap< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::unordered_multimap (initializer_list< pair< _Key, _Tp > >, unordered_multimap< int, int >::size_type, _Allocator) -> unordered_multimap< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  std::unordered_multimap (initializer_list< pair< _Key, _Tp > >, unordered_multimap< int, int >::size_type, _Hash, _Allocator) -> unordered_multimap< _Key, _Tp, _Hash, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  std::unordered_multimap (initializer_list< pair< _Key, _Tp > >, unordered_multimap< int, int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_multimap< _Key, _Tp, _Hash, _Pred, _Allocator >

6.187.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <unordered_map>.

6.188 unordered_set.h File Reference

Classes

- class std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >
- class std::unordered_set< _Value, _Hash, _Pred, _Alloc >

Namespaces

- namespace std

Typedefs

- template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>, typename _Tr = __umset_traits<__cache_default<_Value, _Hash>::value>>
  using **std::__umset_hashtable**
- template<bool _Cache>
  using std::__umset_traits
- template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>, typename _Tr = __uset_traits<__cache_default<_Value, _Hash>::value>>
  using **std::__uset_hashtable**
- template<bool _Cache>
  using std::__uset_traits

Functions

- template<class _Value, class _Hash, class _Pred, class _Alloc>
  bool **std::operator==** (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)
- template<class _Value, class _Hash, class _Pred, class _Alloc>
  bool **std::operator==** (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)
- template<class _Value, class _Hash, class _Pred, class _Alloc>
  void **std::swap** (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))
- template<class _Value, class _Hash, class _Pred, class _Alloc>
  void **std::swap** (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, unordered_set< _Value, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  std::unordered_multiset (_InputIterator, _InputIterator, _Allocator) -> unordered_multiset< typename iterator_traits< _InputIterator >::value_type, hash< typename iterator_traits< _InputIterator >::value_type >, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  std::unordered_multiset (_InputIterator, _InputIterator, unordered_multiset< int >::size_type, _Allocator) -> unordered_multiset< typename iterator_traits< _InputIterator >::value_type, hash< typename iterator_traits< _InputIterator >::value_type >, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  std::unordered_multiset (_InputIterator, _InputIterator, unordered_multiset< int >::size_type, _Hash, _Allocator) -> unordered_multiset< typename iterator_traits< _InputIterator >::value_type, _Hash, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Hash = hash<typename iterator_traits<_InputIterator>::value_type>, typename _Pred = equal_to<typename iterator_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  std::unordered_multiset (_InputIterator, _InputIterator, unordered_multiset< int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_multiset< typename iterator_traits< _InputIterator >::value_type, _Hash, _Pred, _Allocator >
- template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::unordered_multiset (initializer_list< _Tp >, _Allocator) -> unordered_multiset< _Tp, hash< _Tp >, equal_to< _Tp >, _Allocator >

- template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::unordered_multiset (initializer_list< _Tp >, unordered_multiset< int >::size_type, _Allocator) -> unordered_multiset< _Tp, hash< _Tp >, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  std::unordered_multiset (initializer_list< _Tp >, unordered_multiset< int >::size_type, _Hash, _Allocator) -> unordered_multiset< _Tp, _Hash, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Hash = hash<_Tp>, typename _Pred = equal_to<_Tp>, typename _Allocator = allocator<_Tp>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  std::unordered_multiset (initializer_list< _Tp >, unordered_multiset< int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_multiset< _Tp, _Hash, _Pred, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  std::unordered_set (_InputIterator, _InputIterator, _Allocator) -> unordered_set< typename iterator_traits< _InputIterator >::value_type, hash< typename iterator_traits< _InputIterator >::value_type >, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  std::unordered_set (_InputIterator, _InputIterator, unordered_set< int >::size_type, _Allocator) -> unordered_set< typename iterator_traits< _InputIterator >::value_type, hash< typename iterator_traits< _InputIterator >::value_type >, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  std::unordered_set (_InputIterator, _InputIterator, unordered_set< int >::size_type, _Hash, _Allocator) -> unordered_set< typename iterator_traits< _InputIterator >::value_type, _Hash, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Hash = hash<typename iterator_traits<_InputIterator>::value_type>, typename _Pred = equal_to<typename iterator_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  std::unordered_set (_InputIterator, _InputIterator, unordered_set< int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_set< typename iterator_traits< _InputIterator >::value_type, _Hash, _Pred, _Allocator >
- template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::unordered_set (initializer_list< _Tp >, _Allocator) -> unordered_set< _Tp, hash< _Tp >, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::unordered_set (initializer_list< _Tp >, unordered_set< int >::size_type, _Allocator) -> unordered_set< _Tp, hash< _Tp >, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  std::unordered_set (initializer_list< _Tp >, unordered_set< int >::size_type, _Hash, _Allocator) -> unordered_set< _Tp, _Hash, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Hash = hash<_Tp>, typename _Pred = equal_to<_Tp>, typename _Allocator = allocator<_Tp>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  std::unordered_set (initializer_list< _Tp >, unordered_set< int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_set< _Tp, _Hash, _Pred, _Allocator >

6.188.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <unordered_set>.

6.189 uses_allocator_args.h File Reference

Namespaces

- namespace std

Functions

- template<typename _Tp, typename _Alloc, typename... _Args>
  constexpr _Tp **std::make_obj_using_allocator** (const _Alloc &__a, _Args &&... __args)
- template<typename _Tp, typename _Alloc, typename... _Args>
  constexpr _Tp * **std::uninitialized_construct_using_allocator** (_Tp *__p, const _Alloc &__a, _Args &&... _↩
  _args)
- template<_Std_pair _Tp, typename _Alloc>
  constexpr auto **std::uses_allocator_construction_args** (const _Alloc &) noexcept
- template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>
  constexpr auto **std::uses_allocator_construction_args** (const _Alloc &, _Up &&, _Vp &&) noexcept
- template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>
  constexpr auto **std::uses_allocator_construction_args** (const _Alloc &, const pair< _Up, _Vp > &) noexcept
- template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>
  constexpr auto **std::uses_allocator_construction_args** (const _Alloc &, pair< _Up, _Vp > &&) noexcept
- template<typename _Tp, typename _Alloc, typename... _Args>
  requires (! _Std_pair<_Tp>)
  constexpr auto **std::uses_allocator_construction_args** (const _Alloc &__a, _Args &&... __args) noexcept
- template<_Std_pair _Tp, typename _Alloc, typename _Tuple1, typename _Tuple2>
  constexpr auto **std::uses_allocator_construction_args** (const _Alloc &__a, piecewise_construct_t, _Tuple1 &&__x, _Tuple2 &&__y) noexcept

6.189.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

6.190 utility.h File Reference

Classes

- struct std::integer_sequence< _Tp, _Idx >

Namespaces

- namespace std

Typedefs

- template<typename _Tp, typename _Up = typename remove_cv<_Tp>::type, typename = typename enable_if<is_same<_Tp, _Up>↩
  ::value>::type, size_t = tuple_size<_Tp>::value>
  using **std::__enable_if_has_tuple_size**
- template<size_t __i, typename _Tp>
  using **std::__tuple_element_t**
- template<size_t... _Idx>
  using std::index_sequence
- template<typename... _Types>
  using std::index_sequence_for

- template<size_t _Num>
  using std::make_index_sequence
- template<typename _Tp, _Tp _Num>
  using std::make_integer_sequence
- template<size_t __i, typename _Tp>
  using **std::tuple_element_t**

Functions

- template<typename _Tp, typename... _Types>
  constexpr size_t **std::__find_uniq_type_in_pack** ()

Variables

- template<typename>
  constexpr bool **std::__is_in_place_index_v**
- template<size_t _Nm>
  constexpr bool **std::__is_in_place_index_v**< **in_place_index_t**< **_Nm** > >
- template<typename>
  constexpr bool **std::__is_in_place_type_v**
- template<typename _Tp>
  constexpr bool **std::__is_in_place_type_v**< **in_place_type_t**< **_Tp** > >
- template<typename _Range>
  constexpr bool **std::ranges::__detail::__is_subrange**
- constexpr _Swallow_assign std::ignore
- constexpr in_place_t **std::in_place**
- template<size_t _Idx>
  constexpr in_place_index_t< _Idx > **std::in_place_index**
- template<typename _Tp>
  constexpr in_place_type_t< _Tp > **std::in_place_type**
- template<typename _Tp>
  constexpr size_t **std::tuple_size_v**

6.190.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <utility>.
This file contains the parts of `<utility>` needed by other headers, so they don't need to include the whole of `<utility>`.

6.191 valarray_after.h File Reference

Namespaces

- namespace std
- namespace std::__detail

Macros

- #define **_DEFINE_EXPR_BINARY_FUNCTION**(_Fun, _UFun)
- #define **_DEFINE_EXPR_BINARY_OPERATOR**(_Op, _Name)
- #define **_DEFINE_EXPR_UNARY_FUNCTION**(_Name, _UName)
- #define **_DEFINE_EXPR_UNARY_OPERATOR**(_Op, _Name)

Functions

- template<class _Dom>
  _Expr< _UnClos< struct std::_Abs, _Expr, _Dom >, typename _Dom::value_type > **std::abs** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Abs, _ValArray, _Tp >, _Tp > **std::abs** (const valarray< _Tp > &__v)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Acos, _Expr, _Dom >, typename _Dom::value_type > **std::acos** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Acos, _ValArray, _Tp >, _Tp > **std::acos** (const valarray< _Tp > &__v)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Asin, _Expr, _Dom >, typename _Dom::value_type > **std::asin** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Asin, _ValArray, _Tp >, _Tp > **std::asin** (const valarray< _Tp > &__v)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Atan, _Expr, _Dom >, typename _Dom::value_type > **std::atan** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Atan, _ValArray, _Tp >, _Tp > **std::atan** (const valarray< _Tp > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::_Atan2, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _Dom::value_type > **std::atan2** (const _Expr< _Dom, typename _Dom::value_type > &__e, const typename _Dom::value_type &__t)
- template<class _Dom>
  _Expr< _BinClos< struct std::_Atan2, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _Dom::value_type > **std::atan2** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::_Atan2, _Expr, _Expr, _Dom1, _Dom2 >, typename _Dom1::value_type > **std::atan2** (const _Expr< _Dom1, typename _Dom1::value_type > &__e1, const _Expr< _Dom2, typename _Dom2::value_type > &__e2)
- template<class _Dom>
  _Expr< _BinClos< struct std::_Atan2, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type > **std::atan2** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  _Expr< _BinClos< struct std::_Atan2, _Constant, _ValArray, _Tp, _Tp >, _Tp > **std::atan2** (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)
- template<typename _Tp>
  _Expr< _BinClos< struct std::_Atan2, _ValArray, _Constant, _Tp, _Tp >, _Tp > **std::atan2** (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)
- template<typename _Tp>
  _Expr< _BinClos< struct std::_Atan2, _ValArray, _ValArray, _Tp, _Tp >, _Tp > **std::atan2** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::_Atan2, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type > **std::atan2** (const valarray< typename _Dom::valarray > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Cos, _Expr, _Dom >, typename _Dom::value_type > **std::cos** (const _Expr< _Dom, typename _Dom::value_type > &__e)

- template<typename _Tp>
  _Expr< _UnClos< struct std::_Cos, _ValArray, _Tp >, _Tp > **std::cos** (const valarray< _Tp > &__v)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Cosh, _Expr, _Dom >, typename _Dom::value_type > **std::cosh** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Cosh, _ValArray, _Tp >, _Tp > **std::cosh** (const valarray< _Tp > &__v)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Exp, _Expr, _Dom >, typename _Dom::value_type > **std::exp** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Exp, _ValArray, _Tp >, _Tp > **std::exp** (const valarray< _Tp > &__v)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Log, _Expr, _Dom >, typename _Dom::value_type > **std::log** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Log, _ValArray, _Tp >, _Tp > **std::log** (const valarray< _Tp > &__v)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Log10, _Expr, _Dom >, typename _Dom::value_type > **std::log10** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Log10, _ValArray, _Tp >, _Tp > **std::log10** (const valarray< _Tp > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__not_equal_to, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__not_equal_to, typename _Dom::value_type >::result_type > **std::operator!=** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__not_equal_to, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__not_equal_to, typename _Dom::value_type >::result_type > **std::operator!=** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__not_equal_to, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__not_equal_to, typename _Dom1::value_type >::result_type > **std::operator!=** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__not_equal_to, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__not_equal_to, typename _Dom::value_type >::result_type > **std::operator!=** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__not_equal_to, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__not_equal_to, typename _Dom::value_type >::result_type > **std::operator!=** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom>
  _Expr< _BinClos< struct std::__modulus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__modulus, typename _Dom::value_type >::result_type > **std::operator%** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__modulus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__modulus, typename _Dom::value_type >::result_type > **std::operator%** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__modulus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__modulus, typename _Dom1::value_type >::result_type > **std::operator%** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)

- template<class _Dom>
  _Expr< _BinClos< struct std::__modulus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__modulus, typename _Dom::value_type >::result_type > **std::operator%** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__modulus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__modulus, typename _Dom::value_type >::result_type > **std::operator%** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_and, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__bitwise_and, typename _Dom::value_type >::result_type > **std::operator&** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_and, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__bitwise_and, typename _Dom::value_type >::result_type > **std::operator&** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__bitwise_and, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::_↩
  _bitwise_and, typename _Dom1::value_type >::result_type > **std::operator&** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_and, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__bitwise_and, typename _Dom::value_type >::result_type > **std::operator&** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_and, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__bitwise_and, typename _Dom::value_type >::result_type > **std::operator&** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom>
  _Expr< _BinClos< struct std::__logical_and, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__logical_and, typename _Dom::value_type >::result_type > **std::operator&&** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__logical_and, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__logical_and, typename _Dom::value_type >::result_type > **std::operator&&** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__logical_and, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__↩
  logical_and, typename _Dom1::value_type >::result_type > **std::operator&&** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__logical_and, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__logical_and, typename _Dom::value_type >::result_type > **std::operator&&** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__logical_and, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__logical_and, typename _Dom::value_type >::result_type > **std::operator&&** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom>
  _Expr< _BinClos< struct std::__multiplies, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__multiplies, typename _Dom::value_type >::result_type > **std::operator*** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)

- template<class _Dom>
  _Expr< _BinClos< struct std::__multiplies, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__multiplies, typename _Dom::value_type >::result_type > **std::operator*** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__multiplies, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__multiplies, typename _Dom1::value_type >::result_type > **std::operator*** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__multiplies, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__multiplies, typename _Dom::value_type >::result_type > **std::operator*** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__multiplies, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__multiplies, typename _Dom::value_type >::result_type > **std::operator*** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom>
  _Expr< _BinClos< struct std::__plus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__plus, typename _Dom::value_type >::result_type > **std::operator+** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__plus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__plus, typename _Dom::value_type >::result_type > **std::operator+** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__plus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__plus, typename _Dom1::value_type >::result_type > **std::operator+** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__plus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__plus, typename _Dom::value_type >::result_type > **std::operator+** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__plus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__plus, typename _Dom::value_type >::result_type > **std::operator+** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom>
  _Expr< _BinClos< struct std::__minus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__minus, typename _Dom::value_type >::result_type > **std::operator-** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__minus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__minus, typename _Dom::value_type >::result_type > **std::operator-** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__minus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__minus, typename _Dom1::value_type >::result_type > **std::operator-** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__minus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__minus, typename _Dom::value_type >::result_type > **std::operator-** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)

- template<class _Dom>
  _Expr< _BinClos< struct std::__minus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _↩
  _fun< struct std::__minus, typename _Dom::value_type >::result_type > **std::operator-** (const valarray< type-
  name _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom>
  _Expr< _BinClos< struct std::__divides, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _↩
  _fun< struct std::__divides, typename _Dom::value_type >::result_type > **std::operator/** (const _Expr< _Dom,
  typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__divides, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename ↩
  __fun< struct std::__divides, typename _Dom::value_type >::result_type > **std::operator/** (const _Expr< _Dom,
  typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__divides, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__↩
  divides, typename _Dom1::value_type >::result_type > **std::operator/** (const _Expr< _Dom1, typename _↩
  Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__divides, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename
  __fun< struct std::__divides, typename _Dom::value_type >::result_type > **std::operator/** (const typename _↩
  Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__divides, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _↩
  _fun< struct std::__divides, typename _Dom::value_type >::result_type > **std::operator/** (const valarray< type-
  name _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom>
  _Expr< _BinClos< struct std::__less, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _↩
  _fun< struct std::__less, typename _Dom::value_type >::result_type > **std::operator<** (const _Expr< _Dom,
  typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__less, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _↩
  _fun< struct std::__less, typename _Dom::value_type >::result_type > **std::operator<** (const _Expr< _Dom,
  typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__less, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__less,
  typename _Dom1::value_type >::result_type > **std::operator<** (const _Expr< _Dom1, typename _Dom1↩
  ::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__less, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __↩
  fun< struct std::__less, typename _Dom::value_type >::result_type > **std::operator<** (const typename _Dom↩
  ::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__less, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _↩
  _fun< struct std::__less, typename _Dom::value_type >::result_type > **std::operator<** (const valarray< type-
  name _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom>
  _Expr< _BinClos< struct std::__shift_left, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename
  __fun< struct std::__shift_left, typename _Dom::value_type >::result_type > **std::operator<<** (const _Expr<
  _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__shift_left, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename
  __fun< struct std::__shift_left, typename _Dom::value_type >::result_type > **std::operator<<** (const _Expr<
  _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)

- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__shift_left, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::_↩
  _shift_left, typename _Dom1::value_type >::result_type > **std::operator**<< (const _Expr< _Dom1, typename
  _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__shift_left, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename
  __fun< struct std::__shift_left, typename _Dom::value_type >::result_type > **std::operator**<< (const typename
  _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__shift_left, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename
  __fun< struct std::__shift_left, typename _Dom::value_type >::result_type > **std::operator**<< (const valarray<
  typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom>
  _Expr< _BinClos< struct std::__less_equal, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename
  __fun< struct std::__less_equal, typename _Dom::value_type >::result_type > **std::operator**<= (const _Expr<
  _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__less_equal, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename
  __fun< struct std::__less_equal, typename _Dom::value_type >::result_type > **std::operator**<= (const _Expr<
  _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__less_equal, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__↩
  less_equal, typename _Dom1::value_type >::result_type > **std::operator**<= (const _Expr< _Dom1, typename
  _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__less_equal, _Constant, _Expr, typename _Dom::value_type, _Dom >, type-
  name __fun< struct std::__less_equal, typename _Dom::value_type >::result_type > **std::operator**<= (const
  typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__less_equal, _ValArray, _Expr, typename _Dom::value_type, _Dom >, type-
  name __fun< struct std::__less_equal, typename _Dom::value_type >::result_type > **std::operator**<= (const
  valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom>
  _Expr< _BinClos< struct std::__equal_to, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename
  __fun< struct std::__equal_to, typename _Dom::value_type >::result_type > **std::operator==** (const _Expr<
  _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__equal_to, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename
  __fun< struct std::__equal_to, typename _Dom::value_type >::result_type > **std::operator==** (const _Expr<
  _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__equal_to, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::_↩
  _equal_to, typename _Dom1::value_type >::result_type > **std::operator==** (const _Expr< _Dom1, typename
  _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__equal_to, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename
  __fun< struct std::__equal_to, typename _Dom::value_type >::result_type > **std::operator==** (const typename
  _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__equal_to, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename
  __fun< struct std::__equal_to, typename _Dom::value_type >::result_type > **std::operator==** (const valarray<
  typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)

- template<class _Dom>
  _Expr< _BinClos< struct std::__greater, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _↩
  _fun< struct std::__greater, typename _Dom::value_type >::result_type > **std::operator**> (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__greater, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__greater, typename _Dom::value_type >::result_type > **std::operator**> (const _Expr< _↩
  Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__greater, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__↩
  greater, typename _Dom1::value_type >::result_type > **std::operator**> (const _Expr< _Dom1, typename _↩
  Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__greater, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__greater, typename _Dom::value_type >::result_type > **std::operator**> (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__greater, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename ↩
  __fun< struct std::__greater, typename _Dom::value_type >::result_type > **std::operator**> (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom>
  _Expr< _BinClos< struct std::__greater_equal, _Expr, _ValArray, _Dom, typename _Dom::value_type >, type-name __fun< struct std::__greater_equal, typename _Dom::value_type >::result_type > **std::operator**>**=** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__greater_equal, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__greater_equal, typename _Dom::value_type >::result_type > **std::operator**>**=** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__greater_equal, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::↩
  __greater_equal, typename _Dom1::value_type >::result_type > **std::operator**>**=** (const _Expr< _Dom1, type-name _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__greater_equal, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__greater_equal, typename _Dom::value_type >::result_type > **std::operator**>**=** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__greater_equal, _ValArray, _Expr, typename _Dom::value_type, _Dom >, type-name __fun< struct std::__greater_equal, typename _Dom::value_type >::result_type > **std::operator**>**=** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom>
  _Expr< _BinClos< struct std::__shift_right, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__shift_right, typename _Dom::value_type >::result_type > **std::operator**>> (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__shift_right, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__shift_right, typename _Dom::value_type >::result_type > **std::operator**>> (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__shift_right, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__↩
  shift_right, typename _Dom1::value_type >::result_type > **std::operator**>> (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)

- template<class _Dom>
  _Expr< _BinClos< struct std::__shift_right, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__shift_right, typename _Dom::value_type >::result_type > **std::operator**>> (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__shift_right, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__shift_right, typename _Dom::value_type >::result_type > **std::operator**>> (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_xor, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__bitwise_xor, typename _Dom::value_type >::result_type > **std::operator**^ (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_xor, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__bitwise_xor, typename _Dom::value_type >::result_type > **std::operator**^ (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__bitwise_xor, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::_↩ _bitwise_xor, typename _Dom1::value_type >::result_type > **std::operator**^ (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_xor, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__bitwise_xor, typename _Dom::value_type >::result_type > **std::operator**^ (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_xor, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__bitwise_xor, typename _Dom::value_type >::result_type > **std::operator**^ (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_or, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__bitwise_or, typename _Dom::value_type >::result_type > **std::operator**| (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_or, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__bitwise_or, typename _Dom::value_type >::result_type > **std::operator**| (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__bitwise_or, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::↩ __bitwise_or, typename _Dom1::value_type >::result_type > **std::operator**| (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_or, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__bitwise_or, typename _Dom::value_type >::result_type > **std::operator**| (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_or, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__bitwise_or, typename _Dom::value_type >::result_type > **std::operator**| (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom>
  _Expr< _BinClos< struct std::__logical_or, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__logical_or, typename _Dom::value_type >::result_type > **std::operator**|| (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)

- template<class _Dom>
  _Expr< _BinClos< struct std::__logical_or, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__logical_or, typename _Dom::value_type >::result_type > **std::operator||** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__logical_or, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__logical_or, typename _Dom1::value_type >::result_type > **std::operator||** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__logical_or, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__logical_or, typename _Dom::value_type >::result_type > **std::operator||** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__logical_or, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__logical_or, typename _Dom::value_type >::result_type > **std::operator||** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom>
  _Expr< _BinClos< struct std::_Pow, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _Dom::value_type > **std::pow** (const _Expr< _Dom, typename _Dom::value_type > &__e, const typename _Dom::value_type &__t)
- template<class _Dom>
  _Expr< _BinClos< struct std::_Pow, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _Dom::value_type > **std::pow** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::_Pow, _Expr, _Expr, _Dom1, _Dom2 >, typename _Dom1::value_type > **std::pow** (const _Expr< _Dom1, typename _Dom1::value_type > &__e1, const _Expr< _Dom2, typename _Dom2::value_type > &__e2)
- template<class _Dom>
  _Expr< _BinClos< struct std::_Pow, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type > **std::pow** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  _Expr< _BinClos< struct std::_Pow, _Constant, _ValArray, _Tp, _Tp >, _Tp > **std::pow** (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)
- template<typename _Tp>
  _Expr< _BinClos< struct std::_Pow, _ValArray, _Constant, _Tp, _Tp >, _Tp > **std::pow** (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)
- template<typename _Tp>
  _Expr< _BinClos< struct std::_Pow, _ValArray, _ValArray, _Tp, _Tp >, _Tp > **std::pow** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::_Pow, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type > **std::pow** (const valarray< typename _Dom::valarray > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Sin, _Expr, _Dom >, typename _Dom::value_type > **std::sin** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Sin, _ValArray, _Tp >, _Tp > **std::sin** (const valarray< _Tp > &__v)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Sinh, _Expr, _Dom >, typename _Dom::value_type > **std::sinh** (const _Expr< _Dom, typename _Dom::value_type > &__e)

- template<typename _Tp>
  _Expr< _UnClos< struct std::_Sinh, _ValArray, _Tp >, _Tp > **std::sinh** (const valarray< _Tp > &__v)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Sqrt, _Expr, _Dom >, typename _Dom::value_type > **std::sqrt** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Sqrt, _ValArray, _Tp >, _Tp > **std::sqrt** (const valarray< _Tp > &__v)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Tan, _Expr, _Dom >, typename _Dom::value_type > **std::tan** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Tan, _ValArray, _Tp >, _Tp > **std::tan** (const valarray< _Tp > &__v)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Tanh, _Expr, _Dom >, typename _Dom::value_type > **std::tanh** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Tanh, _ValArray, _Tp >, _Tp > **std::tanh** (const valarray< _Tp > &__v)

6.191.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <valarray>.

6.192 valarray_array.h File Reference

Namespaces

- namespace std

Macros

- #define **_DEFINE_ARRAY_FUNCTION**(_Op, _Name)

Functions

- template<typename _Tp>
  void **std::__valarray_copy** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **std::__valarray_copy** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp>
  void **std::__valarray_copy** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp>
  void **std::__valarray_copy** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp>
  void **std::__valarray_copy** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp>
  void **std::__valarray_copy** (_Array< _Tp > __a, size_t __n, size_t __s1, _Array< _Tp > __b, size_t __s2)
- template<typename _Tp>
  void **std::__valarray_copy** (_Array< _Tp > __src, size_t __n, _Array< size_t > __i, _Array< _Tp > __dst, _Array< size_t > __j)
- template<typename _Tp>
  void **std::__valarray_copy** (const _Tp *__restrict__ __a, _Tp *__restrict__ __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **std::__valarray_copy** (const _Tp *__restrict__ __a, const size_t *__restrict__ __i, _Tp *__restrict__ __b, size_t __n)

- template<typename _Tp>
  void **std::__valarray_copy** (const _Tp *__restrict__ __a, size_t __n, _Tp *__restrict__ __b)
- template<typename _Tp>
  void **std::__valarray_copy** (const _Tp *__restrict__ __a, size_t __n, _Tp *__restrict__ __b, const size_t *__restrict__ __i)
- template<typename _Tp>
  void **std::__valarray_copy** (const _Tp *__restrict__ __a, size_t __n, size_t __s, _Tp *__restrict__ __b)
- template<typename _Tp>
  void **std::__valarray_copy** (const _Tp *__restrict__ __src, size_t __n, const size_t *__restrict__ __i, _Tp *__restrict__ __dst, const size_t *__restrict__ __j)
- template<typename _Tp>
  void **std::__valarray_copy** (const _Tp *__restrict__ __src, size_t __n, size_t __s1, _Tp *__restrict__ __dst, size_t __s2)
- template<typename _Tp>
  void **std::__valarray_copy_construct** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp>
  void **std::__valarray_copy_construct** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp>
  void **std::__valarray_copy_construct** (const _Tp *__b, const _Tp *__e, _Tp *__restrict__ __o)
- template<typename _Tp>
  void **std::__valarray_copy_construct** (const _Tp *__restrict__ __a, const size_t *__restrict__ __i, _Tp *__restrict__ __o, size_t __n)
- template<typename _Tp>
  void **std::__valarray_copy_construct** (const _Tp *__restrict__ __a, size_t __n, size_t __s, _Tp *__restrict__ __o)
- template<typename _Tp>
  void **std::__valarray_default_construct** (_Tp *__b, _Tp *__e)
- template<typename _Tp>
  void **std::__valarray_destroy_elements** (_Tp *__b, _Tp *__e)
- template<typename _Tp>
  void **std::__valarray_fill** (_Array< _Tp > __a, _Array< size_t > __i, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **std::__valarray_fill** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **std::__valarray_fill** (_Array< _Tp > __a, size_t __n, size_t __s, const _Tp &__t)
- template<typename _Tp>
  void **std::__valarray_fill** (_Tp *__restrict__ __a, const size_t *__restrict__ __i, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **std::__valarray_fill** (_Tp *__restrict__ __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **std::__valarray_fill** (_Tp *__restrict__ __a, size_t __n, size_t __s, const _Tp &__t)
- template<typename _Tp>
  void **std::__valarray_fill_construct** (_Tp *__b, _Tp *__e, const _Tp __t)
- template<typename _Tp>
  _Tp * **std::__valarray_get_storage** (size_t)
- template<typename _Ta>
  _Ta::value_type **std::__valarray_max** (const _Ta &__a)
- template<typename _Ta>
  _Ta::value_type **std::__valarray_min** (const _Ta &__a)
- void **std::__valarray_release_memory** (void *__p)
- template<typename _Tp>
  _Tp **std::__valarray_sum** (const _Tp *__f, const _Tp *__l)

- template<typename _Tp>
  void **std::_Array_augmented___bitwise_and** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_and** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___bitwise_and** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_and** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___bitwise_and** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___bitwise_and** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_and** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_and** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_and** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_and** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_and** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___bitwise_and** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_or** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_or** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___bitwise_or** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_or** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___bitwise_or** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___bitwise_or** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_or** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_or** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)

- template<typename _Tp>
  void **std::_Array_augmented___bitwise_or** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_or** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_or** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___bitwise_or** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &↩ __e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_xor** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_xor** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___bitwise_xor** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_xor** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___bitwise_xor** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___bitwise_xor** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_xor** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_xor** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_xor** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_xor** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_xor** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___bitwise_xor** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___divides** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **std::_Array_augmented___divides** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size↩ _t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___divides** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___divides** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)

- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___divides** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___divides** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___divides** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp>
  void **std::_Array_augmented___divides** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp>
  void **std::_Array_augmented___divides** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp>
  void **std::_Array_augmented___divides** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **std::_Array_augmented___divides** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___divides** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___minus** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **std::_Array_augmented___minus** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___minus** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___minus** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___minus** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___minus** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___minus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp>
  void **std::_Array_augmented___minus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp>
  void **std::_Array_augmented___minus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp>
  void **std::_Array_augmented___minus** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **std::_Array_augmented___minus** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___minus** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___modulus** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)

- template<typename _Tp>
  void **std::_Array_augmented___modulus** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___modulus** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___modulus** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___modulus** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___modulus** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___modulus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp>
  void **std::_Array_augmented___modulus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp>
  void **std::_Array_augmented___modulus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp>
  void **std::_Array_augmented___modulus** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **std::_Array_augmented___modulus** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___modulus** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___multiplies** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **std::_Array_augmented___multiplies** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___multiplies** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___multiplies** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___multiplies** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___multiplies** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___multiplies** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp>
  void **std::_Array_augmented___multiplies** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp>
  void **std::_Array_augmented___multiplies** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)

- template<typename _Tp>
  void **std::_Array_augmented___multiplies** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **std::_Array_augmented___multiplies** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___multiplies** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___plus** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **std::_Array_augmented___plus** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___plus** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___plus** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___plus** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___plus** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___plus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp>
  void **std::_Array_augmented___plus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp>
  void **std::_Array_augmented___plus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp>
  void **std::_Array_augmented___plus** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **std::_Array_augmented___plus** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___plus** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___shift_left** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **std::_Array_augmented___shift_left** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___shift_left** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___shift_left** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___shift_left** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)

- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___shift_left** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___shift_left** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp>
  void **std::_Array_augmented___shift_left** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp>
  void **std::_Array_augmented___shift_left** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp>
  void **std::_Array_augmented___shift_left** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **std::_Array_augmented___shift_left** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___shift_left** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___shift_right** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **std::_Array_augmented___shift_right** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___shift_right** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___shift_right** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___shift_right** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___shift_right** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___shift_right** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp>
  void **std::_Array_augmented___shift_right** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp>
  void **std::_Array_augmented___shift_right** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp>
  void **std::_Array_augmented___shift_right** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **std::_Array_augmented___shift_right** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___shift_right** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)

6.192.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <valarray>.

6.193 valarray_array.tcc File Reference

Namespaces

- namespace std

Macros

- #define **_VALARRAY_ARRAY_TCC**

Functions

- template<typename _Tp>
  void **std::__valarray_copy** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp>
  void **std::__valarray_copy** (_Array< _Tp > __a, _Array< bool > __m, size_t __n, _Array< _Tp > __b, _Array< bool > __k)
- template<typename _Tp>
  void **std::__valarray_copy** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp>
  void **std::__valarray_copy** (_Array< _Tp > __e, _Array< size_t > __f, size_t __n, _Array< _Tp > __a, _Array< size_t > __i)
- template<typename _Tp, class _Dom>
  void **std::__valarray_copy** (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a)
- template<typename _Tp, class _Dom>
  void **std::__valarray_copy** (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, _Array< bool > __m)
- template<typename _Tp, class _Dom>
  void **std::__valarray_copy** (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, _Array< size_t > __i)
- template<typename _Tp, class _Dom>
  void **std::__valarray_copy** (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, size_t __s)
- template<typename _Tp>
  void **std::__valarray_copy_construct** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::__valarray_copy_construct** (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a)
- template<typename _Tp>
  void **std::__valarray_fill** (_Array< _Tp > __a, size_t __n, _Array< bool > __m, const _Tp &__t)

6.193.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <valarray>.

6.194 valarray_before.h File Reference

Namespaces

- namespace std
- namespace std::__detail

6.194.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <valarray>.

6.195 vector.tcc File Reference

Namespaces

- namespace std

Macros

- #define **_VECTOR_TCC**

6.195.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <vector>.

6.196 version.h File Reference

Macros

- #define **__glibcxx_addressof_constexpr**
- #define **__glibcxx_allocator_traits_is_always_equal**
- #define **__glibcxx_any**
- #define **__glibcxx_apply**
- #define **__glibcxx_array_constexpr**
- #define **__glibcxx_as_const**
- #define **__glibcxx_assume_aligned**
- #define **__glibcxx_atomic_flag_test**
- #define **__glibcxx_atomic_float**
- #define **__glibcxx_atomic_is_always_lock_free**
- #define **__glibcxx_atomic_ref**
- #define **__glibcxx_atomic_shared_ptr**
- #define **__glibcxx_atomic_value_initialization**
- #define **__glibcxx_atomic_wait**
- #define **__glibcxx_barrier**
- #define **__glibcxx_bind_front**
- #define **__glibcxx_bit_cast**
- #define **__glibcxx_bitops**
- #define **__glibcxx_bool_constant**
- #define **__glibcxx_bounded_array_traits**
- #define **__glibcxx_boyer_moore_searcher**
- #define **__glibcxx_byte**
- #define **__glibcxx_chrono**
- #define **__glibcxx_chrono_cxx20**
- #define **__glibcxx_chrono_udls**
- #define **__glibcxx_clamp**
- #define **__glibcxx_complex_udls**
- #define **__glibcxx_concepts**
- #define **__glibcxx_constexpr_algorithms**
- #define **__glibcxx_constexpr_char_traits**
- #define **__glibcxx_constexpr_complex**
- #define **__glibcxx_constexpr_dynamic_alloc**
- #define **__glibcxx_constexpr_functional**
- #define **__glibcxx_constexpr_iterator**

- #define **__glibcxx_constexpr_memory**
- #define **__glibcxx_constexpr_numeric**
- #define **__glibcxx_constexpr_string**
- #define **__glibcxx_constexpr_string_view**
- #define **__glibcxx_constexpr_tuple**
- #define **__glibcxx_constexpr_utility**
- #define **__glibcxx_constexpr_vector**
- #define **__glibcxx_constrained_equality**
- #define **__glibcxx_coroutine**
- #define **__glibcxx_enable_shared_from_this**
- #define **__glibcxx_endian**
- #define **__glibcxx_erase_if**
- #define **__glibcxx_exchange_function**
- #define **__glibcxx_execution**
- #define **__glibcxx_filesystem**
- #define **__glibcxx_format**
- #define **__glibcxx_format_uchar**
- #define **__glibcxx_gcd**
- #define **__glibcxx_gcd_lcm**
- #define **__glibcxx_generic_associative_lookup**
- #define **__glibcxx_generic_unordered_lookup**
- #define **__glibcxx_has_unique_object_representations**
- #define **__glibcxx_hypot**
- #define **__glibcxx_incomplete_container_elements**
- #define **__glibcxx_int_pow2**
- #define **__glibcxx_integer_comparison_functions**
- #define **__glibcxx_integer_sequence**
- #define **__glibcxx_integral_constant_callable**
- #define **__glibcxx_interpolate**
- #define **__glibcxx_invoke**
- #define **__glibcxx_is_aggregate**
- #define **__glibcxx_is_constant_evaluated**
- #define **__glibcxx_is_final**
- #define **__glibcxx_is_invocable**
- #define **__glibcxx_is_layout_compatible**
- #define **__glibcxx_is_nothrow_convertible**
- #define **__glibcxx_is_null_pointer**
- #define **__glibcxx_is_pointer_interconvertible**
- #define **__glibcxx_is_swappable**
- #define **__glibcxx_jthread**
- #define **__glibcxx_latch**
- #define **__glibcxx_launder**
- #define **__glibcxx_lcm**
- #define **__glibcxx_list_remove_return_type**
- #define **__glibcxx_logical_traits**
- #define **__glibcxx_make_from_tuple**
- #define **__glibcxx_make_obj_using_allocator**
- #define **__glibcxx_make_reverse_iterator**
- #define **__glibcxx_make_unique**
- #define **__glibcxx_map_try_emplace**
- #define **__glibcxx_math_constants**

- #define **__glibcxx_math_spec_funcs**
- #define **__glibcxx_math_special_functions**
- #define **__glibcxx_memory_resource**
- #define **__glibcxx_move_iterator_concept**
- #define **__glibcxx_node_extract**
- #define **__glibcxx_nonmember_container_access**
- #define **__glibcxx_not_fn**
- #define **__glibcxx_null_iterators**
- #define **__glibcxx_optional**
- #define **__glibcxx_parallel_algorithm**
- #define **__glibcxx_polymorphic_allocator**
- #define **__glibcxx_quoted_string_io**
- #define **__glibcxx_ranges**
- #define **__glibcxx_raw_memory_algorithms**
- #define **__glibcxx_remove_cvref**
- #define **__glibcxx_result_of_sfinae**
- #define **__glibcxx_robust_nonmodifying_seq_ops**
- #define **__glibcxx_sample**
- #define **__glibcxx_scoped_lock**
- #define **__glibcxx_semaphore**
- #define **__glibcxx_shared_mutex**
- #define **__glibcxx_shared_ptr_arrays**
- #define **__glibcxx_shared_ptr_weak_type**
- #define **__glibcxx_shared_timed_mutex**
- #define **__glibcxx_shift**
- #define **__glibcxx_smart_ptr_for_overwrite**
- #define **__glibcxx_source_location**
- #define **__glibcxx_span**
- #define **__glibcxx_ssize**
- #define **__glibcxx_starts_ends_with**
- #define **__glibcxx_string_udls**
- #define **__glibcxx_string_view**
- #define **__glibcxx_three_way_comparison**
- #define **__glibcxx_to_address**
- #define **__glibcxx_transformation_trait_aliases**
- #define **__glibcxx_transparent_operators**
- #define **__glibcxx_tuple_element_t**
- #define **__glibcxx_tuples_by_type**
- #define **__glibcxx_type_identity**
- #define **__glibcxx_type_trait_variable_templates**
- #define **__glibcxx_uncaught_exceptions**
- #define **__glibcxx_unordered_map_try_emplace**
- #define **__glibcxx_unwrap_ref**
- #define **__glibcxx_variant**
- #define **__glibcxx_void_t**

6.196.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <version>.

6.197 bitset File Reference

Classes

- struct std::_Base_bitset< _Nw >
- struct std::_Base_bitset< 0 >
- struct std::_Base_bitset< 1 >
- class std::bitset< _Nb >
- struct std::hash<::bitset< _Nb > >
- class std::bitset< _Nb >::reference

Namespaces

- namespace std

Macros

- #define **__glibcxx_want_bitset**
- #define **__glibcxx_want_constexpr_bitset**
- #define **_GLIBCXX_BITSET**
- #define **_GLIBCXX_BITSET_BITS_PER_ULL**
- #define **_GLIBCXX_BITSET_BITS_PER_WORD**
- #define **_GLIBCXX_BITSET_WORDS**(__n)

Typedefs

- template<typename _CharT>
  using **std::__bitset::__string**

Functions

- template<size_t _Nb>
  constexpr bitset< _Nb > std::operator& (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept
- template<size_t _Nb>
  constexpr bitset< _Nb > std::operator^ (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept
- template<size_t _Nb>
  constexpr bitset< _Nb > std::operator| (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept

- template<class _CharT, class _Traits, size_t _Nb>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const bitset< _Nb > &__x)
- template<class _CharT, class _Traits, size_t _Nb>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, bitset< _Nb > &__x)

6.197.1 Detailed Description

This is a Standard C++ Library header.

6.198 debug/bitset File Reference

Classes

- class std::__debug::bitset< _Nb >
- struct std::hash< __debug::bitset< _Nb > >

Namespaces

- namespace std
- namespace std::__debug

Functions

- template<size_t _Nb>
  constexpr bitset< _Nb > **std::__debug::operator&** (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept
- template<typename _CharT, typename _Traits, size_t _Nb>
  std::basic_ostream< _CharT, _Traits > & **std::__debug::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const bitset< _Nb > &__x)
- template<typename _CharT, typename _Traits, size_t _Nb>
  std::basic_istream< _CharT, _Traits > & **std::__debug::operator>>** (std::basic_istream< _CharT, _Traits > &__is, bitset< _Nb > &__x)
- template<size_t _Nb>
  constexpr bitset< _Nb > **std::__debug::operator^** (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept
- template<size_t _Nb>
  constexpr bitset< _Nb > **std::__debug::operator|** (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept

6.198.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

6.199 cassert File Reference

6.199.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the *.h implementation files.
This is the C++ version of the Standard C Library header `assert.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

6.200 ccomplex File Reference

Macros

- #define **_GLIBCXX_CCOMPLEX**

6.200.1 Detailed Description

This is a Standard C++ Library header.

Since

C++11 (removed in C++20)

6.201 tr1/ccomplex File Reference

Macros

- #define **_GLIBCXX_TR1_CCOMPLEX**

6.201.1 Detailed Description

This is a TR1 C++ Library header.

6.202 cctype File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CCTYPE**

6.202.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*`.h implementation files.
This is the C++ version of the Standard C Library header `ctype.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

6.203 tr1/cctype File Reference

Macros

- #define **_GLIBCXX_TR1_CCTYPE**

6.203.1 Detailed Description

This is a TR1 C++ Library header.

6.204 cerrno File Reference

Macros

- #define **_GLIBCXX_CERRNO**
- #define **errno**

6.204.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*`.h implementation files.
This is the C++ version of the Standard C Library header `errno.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

6.205 cfenv File Reference

Macros

- #define **_GLIBCXX_CFENV**

6.205.1 Detailed Description

This is a Standard C++ Library header.

6.206 tr1/cfenv File Reference

Macros

- #define **_GLIBCXX_TR1_CFENV**

6.206.1 Detailed Description

This is a TR1 C++ Library header.

6.207 cfloat File Reference

Macros

- #define **_GLIBCXX_CFLOAT**
- #define **DECIMAL_DIG**
- #define **FLT_EVAL_METHOD**

6.207.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*`.h implementation files.
This is the C++ version of the Standard C Library header `float.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

6.208 tr1/cfloat File Reference

Macros

- #define **_GLIBCXX_TR1_CFLOAT**

6.208.1 Detailed Description

This is a TR1 C++ Library header.

6.209 charconv File Reference

Classes

- struct std::from_chars_result
- struct std::to_chars_result

Namespaces

- namespace std
- namespace std::__detail

Macros

- #define **__glibcxx_want_constexpr_charconv**
- #define **__glibcxx_want_to_chars**
- #define **_GLIBCXX_CHARCONV**
- #define **_GLIBCXX_TO_CHARS**(T)

Typedefs

- template<typename _Tp>
  using **std::__detail::__unsigned_least_t**

Enumerations

- enum class std::chars_format { **scientific** , **fixed** , **hex** , **general** }

Functions

- template<bool _DecOnly, typename _Tp>
  constexpr bool std::__detail::__from_chars_alnum (const char *&__first, const char *__last, _Tp &__val, int __base)
- template<bool _DecOnly = false>
  constexpr unsigned char **std::__detail::__from_chars_alnum_to_val** (unsigned char __c)
- from_chars_result **std::__from_chars_bfloat16_t** (const char *__first, const char *__last, float &__value, chars_format __fmt=chars_format::general) noexcept
- from_chars_result **std::__from_chars_float16_t** (const char *__first, const char *__last, float &__value, chars_format __fmt=chars_format::general) noexcept
- template<bool _DecOnly, typename _Tp>
  constexpr bool std::__detail::__from_chars_pow2_base (const char *&__first, const char *__last, _Tp &__val, int __base)
- template<typename _Tp>
  constexpr bool **std::__detail::__raise_and_add** (_Tp &__val, int __base, unsigned char __c)
- template<typename _Tp>
  constexpr to_chars_result **std::__detail::__to_chars** (char *__first, char *__last, _Tp __val, int __base) noexcept
- template<typename _Tp>
  constexpr to_chars_result **std::__detail::__to_chars_10** (char *__first, char *__last, _Tp __val) noexcept
- template<typename _Tp>
  constexpr to_chars_result **std::__detail::__to_chars_16** (char *__first, char *__last, _Tp __val) noexcept
- template<typename _Tp>
  constexpr to_chars_result **std::__detail::__to_chars_2** (char *__first, char *__last, _Tp __val) noexcept
- template<typename _Tp>
  constexpr to_chars_result **std::__detail::__to_chars_8** (char *__first, char *__last, _Tp __val) noexcept
- template<typename _Tp>
  constexpr to_chars_result **std::__to_chars_i** (char *__first, char *__last, _Tp __value, int __base=10)
- template<typename _Tp>
  constexpr unsigned **std::__detail::__to_chars_len** (_Tp __value, int __base) noexcept
- template<typename _Tp>
  constexpr unsigned **std::__detail::__to_chars_len_2** (_Tp __value) noexcept
- template<typename _Tp, enable_if_t< __or_< __is_standard_integer< _Tp >, is_same< char, remove_cv_t< _Tp > > >::value, int > = 0>
  constexpr from_chars_result std::from_chars (const char *__first, const char *__last, _Tp &__value, int __base=10)
- from_chars_result **std::from_chars** (const char *__first, const char *__last, double &__value, chars_format __fmt=chars_format::general) noexcept
- from_chars_result **std::from_chars** (const char *__first, const char *__last, float &__value, chars_format __fmt=chars_format::general) noexcept
- from_chars_result **std::from_chars** (const char *__first, const char *__last, long double &__value, chars_format __fmt=chars_format::general) noexcept
- constexpr chars_format **std::operator&** (chars_format __lhs, chars_format __rhs) noexcept
- constexpr chars_format & **std::operator&=** (chars_format &__lhs, chars_format __rhs) noexcept

- constexpr chars_format **std::operator^** (chars_format __lhs, chars_format __rhs) noexcept
- constexpr chars_format & **std::operator^=** (chars_format &__lhs, chars_format __rhs) noexcept
- constexpr chars_format **std::operator|** (chars_format __lhs, chars_format __rhs) noexcept
- constexpr chars_format & **std::operator|=** (chars_format &__lhs, chars_format __rhs) noexcept
- constexpr chars_format **std::operator~** (chars_format __fmt) noexcept
- to_chars_result **std::to_chars** (char *, char *, bool, int=10)=delete
- constexpr to_chars_result **std::to_chars** (char *__first, char *__last, char __value, int __base=10)
- constexpr to_chars_result **std::to_chars** (char *__first, char *__last, signed char __value, int __base=10)
- constexpr to_chars_result **std::to_chars** (char *__first, char *__last, signed int __value, int __base=10)
- constexpr to_chars_result **std::to_chars** (char *__first, char *__last, signed long __value, int __base=10)
- constexpr to_chars_result **std::to_chars** (char *__first, char *__last, signed long long __value, int __base=10)
- constexpr to_chars_result **std::to_chars** (char *__first, char *__last, signed short __value, int __base=10)
- constexpr to_chars_result **std::to_chars** (char *__first, char *__last, unsigned char __value, int __base=10)
- constexpr to_chars_result **std::to_chars** (char *__first, char *__last, unsigned int __value, int __base=10)
- constexpr to_chars_result **std::to_chars** (char *__first, char *__last, unsigned long __value, int __base=10)
- constexpr to_chars_result **std::to_chars** (char *__first, char *__last, unsigned long long __value, int __base=10)
- constexpr to_chars_result **std::to_chars** (char *__first, char *__last, unsigned short __value, int __base=10)

6.209.1 Detailed Description

This is a Standard C++ Library header.

6.210 chrono File Reference

Classes

- class std::chrono::tzdb_list::const_iterator
- class std::chrono::gps_clock
- class std::chrono::hh_mm_ss< _Duration >
- class std::chrono::tai_clock
- class std::chrono::tzdb_list
- class std::chrono::utc_clock

Namespaces

- namespace std
- namespace std::chrono
- namespace std::literals
- namespace std::literals::chrono_literals

Macros

- #define **__glibcxx_want_chrono**
- #define **__glibcxx_want_chrono_udls**
- #define **_GLIBCXX_CHRONO**

Typedefs

- using **std::chrono::__detail::__months_years_conversion_disambiguator**
- using **std::chrono::gps_seconds**
- template<typename _Duration>
  using **std::chrono::gps_time**
- using **std::chrono::local_days**
- using **std::chrono::local_seconds**
- template<typename _Duration>
  using **std::chrono::local_time**
- using **std::chrono::tai_seconds**
- template<typename _Duration>
  using **std::chrono::tai_time**
- using **std::chrono::utc_seconds**
- template<typename _Duration>
  using **std::chrono::utc_time**
- using **std::chrono::zoned_seconds**

Enumerations

- enum class **choose** { **earliest** , **latest** }

Functions

- template<unsigned __d, typename _Tp>
  constexpr unsigned **std::chrono::__detail::__add_modulo** (unsigned __x, _Tp __y)
- leap_second_info **std::chrono::__detail::__get_leap_second_info** (sys_seconds __ss, bool __is_utc)
- template<unsigned __d, typename _Tp>
  consteval auto **std::chrono::__detail::__modulo_offset** ()
- template<unsigned __d, typename _Tp>
  constexpr unsigned **std::chrono::__detail::__sub_modulo** (unsigned __x, _Tp __y)
- template<typename _Duration>
  void **std::chrono::__throw_bad_local_time** (const local_time< _Duration > &__tp, const local_info &__i)
- template<typename _DestClock, typename _SourceClock, typename _Duration>
  requires __detail::__clock_convs<_DestClock, _SourceClock, _Duration> || __detail::__clock_convs_sys<_DestClock, _SourceClock, ↩ _Duration> || __detail::__clock_convs_utc<_DestClock, _SourceClock, _Duration> || __detail::__clock_convs_sys_utc<_DestClock, _↩ SourceClock, _Duration> || __detail::__clock_convs_utc_sys<_DestClock, _SourceClock, _Duration>
  auto std::chrono::clock_cast (const time_point< _SourceClock, _Duration > &__t)
- const time_zone * **std::chrono::current_zone** ()
- template<typename _Duration>
  leap_second_info **std::chrono::get_leap_second_info** (const utc_time< _Duration > &__ut)
- const tzdb & **std::chrono::get_tzdb** ()
- tzdb_list & **std::chrono::get_tzdb_list** ()
- constexpr bool **std::chrono::is_am** (const hours &__h) noexcept
- constexpr bool **std::chrono::is_pm** (const hours &__h) noexcept
- const time_zone * **std::chrono::locate_zone** (string_view __tz_name)
- constexpr hours **std::chrono::make12** (const hours &__h) noexcept
- constexpr hours **std::chrono::make24** (const hours &__h, bool __is_pm) noexcept
- constexpr chrono::day std::literals::chrono_literals::operator""d (unsigned long long __d) noexcept
- constexpr chrono::year std::literals::chrono_literals::operator""y (unsigned long long __y) noexcept
- template<typename _Dur1, typename _TZPtr1, typename _Dur2, typename _TZPtr2>
  bool **std::chrono::operator==** (const zoned_time< _Dur1, _TZPtr1 > &__x, const zoned_time< _Dur2, _TZPtr2 > &__y)

- const tzdb & **std::chrono::reload_tzdb** ()
- string **std::chrono::remote_version** ()
- **std::chrono::zoned_time** () -> zoned_time< seconds >
- template<typename _TimeZonePtrOrName>
  std::chrono::zoned_time (_TimeZonePtrOrName &&) -> zoned_time< seconds, __time_zone_representation< _TimeZonePtrOrName > >
- template<typename _TimeZonePtrOrName, typename _Duration>
  std::chrono::zoned_time (_TimeZonePtrOrName &&, local_time< _Duration >, choose=choose::earliest) -> zoned_time< common_type_t< _Duration, seconds >, __time_zone_representation< _TimeZonePtrOrName > >
- template<typename _TimeZonePtrOrName, typename _Duration>
  std::chrono::zoned_time (_TimeZonePtrOrName &&, sys_time< _Duration >) -> zoned_time< common_type_t< _Duration, seconds >, __time_zone_representation< _TimeZonePtrOrName > >
- template<typename _Duration, typename _TimeZonePtrOrName, typename _TimeZonePtr2>
  std::chrono::zoned_time (_TimeZonePtrOrName &&, zoned_time< _Duration, _TimeZonePtr2 >, choose=choose↩ ::earliest) -> zoned_time< common_type_t< _Duration, seconds >, __time_zone_representation< _Time↩ ZonePtrOrName > >
- template<typename _Duration>
  std::chrono::zoned_time (sys_time< _Duration >) -> zoned_time< common_type_t< _Duration, seconds > >

Variables

- constexpr unsigned **std::chrono::__detail::__days_per_month** [12]
- template<typename _Tp, typename _Clock>
  constexpr bool **std::chrono::__is_time_point_for_v**
- template<typename _Clock, typename _Duration>
  constexpr bool **std::chrono::__is_time_point_for_v**< **time_point**< **_Clock, _Duration** >, **_Clock** >
- constexpr month **std::chrono::April**
- constexpr month **std::chrono::August**
- constexpr month **std::chrono::December**
- constexpr month **std::chrono::February**
- constexpr weekday **std::chrono::Friday**
- template<> constexpr bool **std::chrono::is_clock_v**< **gps_clock** >
- template<> constexpr bool **std::chrono::is_clock_v**< **tai_clock** >
- template<> constexpr bool **std::chrono::is_clock_v**< **utc_clock** >
- constexpr month **std::chrono::January**
- constexpr month **std::chrono::July**
- constexpr month **std::chrono::June**
- constexpr last_spec **std::chrono::last**
- constexpr month **std::chrono::March**
- constexpr month **std::chrono::May**
- constexpr weekday **std::chrono::Monday**
- constexpr month **std::chrono::November**
- constexpr month **std::chrono::October**
- constexpr weekday **std::chrono::Saturday**
- constexpr month **std::chrono::September**
- constexpr weekday **std::chrono::Sunday**
- constexpr weekday **std::chrono::Thursday**
- constexpr weekday **std::chrono::Tuesday**
- constexpr weekday **std::chrono::Wednesday**

6.210.1 Detailed Description

This is a Standard C++ Library header.

6.211 experimental/chrono File Reference

Namespaces

- namespace std
- namespace std::chrono

Macros

- #define **_GLIBCXX_EXPERIMENTAL_CHRONO**

Variables

- template<typename _Rep>
  constexpr bool **std::chrono::experimental::treat_as_floating_point_v**

6.211.1 Detailed Description

This is a TS C++ Library header.

6.212 cinttypes File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CINTTYPES**

6.212.1 Detailed Description

This is a Standard C++ Library header.

6.213 tr1/cinttypes File Reference

Macros

- #define **_GLIBCXX_TR1_CINTTYPES**

6.213.1 Detailed Description

This is a TR1 C++ Library header.

6.214 ciso646 File Reference

6.214.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the *.h implementation files.
This is the C++ version of the Standard C Library header `iso646.h`, which is empty in C++.

Since

C++11 (removed in C++20)

6.215 climits File Reference

Macros

- #define **_GLIBCXX_CLIMITS**
- #define **LLONG_MAX**
- #define **LLONG_MIN**
- #define **ULLONG_MAX**

6.215.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*`.h implementation files.
This is the C++ version of the Standard C Library header `limits.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

6.216 tr1/climits File Reference

Macros

- #define **_GLIBCXX_TR1_CLIMITS**

6.216.1 Detailed Description

This is a TR1 C++ Library header.

6.217 clocale File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CLOCALE**

6.217.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*`.h implementation files.
This is the C++ version of the Standard C Library header `locale.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

6.218 cmath File Reference

Namespaces

- namespace std

Macros

- #define **__glibcxx_want_hypot**
- #define **__glibcxx_want_interpolate**
- #define **_GLIBCXX_CMATH**
- #define **_GLIBCXX_INCLUDE_NEXT_C_HEADERS**

Functions

- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::acos** (_Tp __x)
- constexpr float **std::acos** (float __x)
- constexpr long double **std::acos** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::acos(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::acos** (simd< _Tp, _Abi > __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::acosh** (_Tp __x)
- constexpr float std::acosh (float __x)
- constexpr long double **std::acosh** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::acosh(declval<double>())), ↩ _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::acosh** (simd< _Tp, _Abi > __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::asin** (_Tp __x)
- constexpr float **std::asin** (float __x)
- constexpr long double **std::asin** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::asin(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::asin** (simd< _Tp, ↩ _Abi > __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::asinh** (_Tp __x)
- constexpr float **std::asinh** (float __x)
- constexpr long double **std::asinh** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::asinh(declval<double>())), ↩ _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::asinh** (simd< _Tp, _Abi > __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::atan** (_Tp __x)
- constexpr float **std::atan** (float __x)
- constexpr long double **std::atan** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::atan(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::atan** (simd< _Tp, ↩ _Abi > __x)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type **std::atan2** (_Tp __y, _Up __x)

- template<typename _Tp, typename _Abi, typename..., typename _Arg2 = _Extra_argument_type<_Tp, _Tp, _Abi>, typename _R = _↩ Math_return_type_t< decltype(std::atan2(declval<double>(), _Arg2::declval())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::atan2** (const simd< _Tp, _Abi > &__x, const typename _Arg2::type &__y)
- constexpr float **std::atan2** (float __y, float __x)
- constexpr long double **std::atan2** (long double __y, long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::atanh** (_Tp __x)
- constexpr float **std::atanh** (float __x)
- constexpr long double **std::atanh** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::atanh(declval<double>())), ↩ _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::atanh** (simd< _Tp, _Abi > __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::cbrt** (_Tp __x)
- constexpr float **std::cbrt** (float __x)
- constexpr long double **std::cbrt** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::cbrt(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::cbrt** (simd< _Tp, _Abi > __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::ceil** (_Tp __x)
- constexpr float **std::ceil** (float __x)
- constexpr long double **std::ceil** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::ceil(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::ceil** (simd< _Tp, _Abi > __x)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type **std::copysign** (_Tp __x, _Up __y)
- template<typename _Tp, typename _Abi, typename = __detail::__odr_helper>
  enable_if_t< is_floating_point_v< _Tp >, simd< _Tp, _Abi > > **std::copysign** (const simd< _Tp, _Abi > &__x, const simd< _Tp, _Abi > &__y)
- constexpr float **std::copysign** (float __x, float __y)
- constexpr long double **std::copysign** (long double __x, long double __y)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::cos** (_Tp __x)
- template<typename _Tp, typename _Abi, typename = __detail::__odr_helper>
  enable_if_t< is_floating_point_v< _Tp >, simd< _Tp, _Abi > > **std::cos** (const simd< _Tp, _Abi > &__x)
- constexpr float **std::cos** (float __x)
- constexpr long double **std::cos** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::cosh** (_Tp __x)
- constexpr float **std::cosh** (float __x)
- constexpr long double **std::cosh** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::cosh(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::cosh** (simd< _Tp, _Abi > __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::erf** (_Tp __x)

- constexpr float **std::erf** (float __x)
- constexpr long double **std::erf** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::erf(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::erf** (simd< _Tp, _Abi > __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::erfc** (_Tp __x)
- constexpr float **std::erfc** (float __x)
- constexpr long double **std::erfc** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::erfc(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::erfc** (simd< _Tp, _Abi > __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::exp** (_Tp __x)
- constexpr float **std::exp** (float __x)
- constexpr long double **std::exp** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::exp(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::exp** (simd< _Tp, _Abi > __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::exp2** (_Tp __x)
- constexpr float **std::exp2** (float __x)
- constexpr long double **std::exp2** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::exp2(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::exp2** (simd< _Tp, _Abi > __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::expm1** (_Tp __x)
- constexpr float **std::expm1** (float __x)
- constexpr long double **std::expm1** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::expm1(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::expm1** (simd< _Tp, _Abi > __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::fabs** (_Tp __x)
- constexpr float **std::fabs** (float __x)
- constexpr long double **std::fabs** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::fabs(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::fabs** (simd< _Tp, ↩ _Abi > __x)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type **std::fdim** (_Tp __x, _Up __y)
- template<typename _Tp, typename _Abi, typename..., typename _Arg2 = _Extra_argument_type<_Tp, _Tp, _Abi>, typename _R = _↩ Math_return_type_t< decltype(std::fdim(declval<double>(), _Arg2::declval())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::fdim** (const simd< _Tp, _Abi > &__x, const typename _Arg2::type &__y)

- constexpr float **std::fdim** (float __x, float __y)
- constexpr long double **std::fdim** (long double __x, long double __y)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::floor** (_Tp __x)
- constexpr float **std::floor** (float __x)
- constexpr long double **std::floor** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::floor(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::floor** (simd< _Tp, _Abi > __x)
- template<typename _Tp, typename _Up, typename _Vp>
  constexpr __gnu_cxx::__promote_3< _Tp, _Up, _Vp >::__type **std::fma** (_Tp __x, _Up __y, _Vp __z)
- template<typename _Tp, typename _Abi, typename..., typename _Arg2 = _Extra_argument_type<_Tp, _Tp, _Abi>, typename _Arg3 = _Extra_argument_type<_Tp, _Tp, _Abi>, typename _R = _Math_return_type_t< decltype(std::fma(declval<double>(), _Arg2::declval(), _Arg3::declval())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::fma** (const simd< _Tp, _Abi > &__x, const typename _Arg2::type &__y, const typename _Arg3::type &__z)
- constexpr float **std::fma** (float __x, float __y, float __z)
- constexpr long double **std::fma** (long double __x, long double __y, long double __z)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type **std::fmax** (_Tp __x, _Up __y)
- template<typename _Tp, typename _Abi, typename..., typename _Arg2 = _Extra_argument_type<_Tp, _Tp, _Abi>, typename _R = _↩ Math_return_type_t< decltype(std::fmax(declval<double>(), _Arg2::declval())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::fmax** (const simd< _Tp, _Abi > &__x, const typename _Arg2::type &__y)
- constexpr float **std::fmax** (float __x, float __y)
- constexpr long double **std::fmax** (long double __x, long double __y)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type **std::fmin** (_Tp __x, _Up __y)
- template<typename _Tp, typename _Abi, typename..., typename _Arg2 = _Extra_argument_type<_Tp, _Tp, _Abi>, typename _R = _↩ Math_return_type_t< decltype(std::fmin(declval<double>(), _Arg2::declval())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::fmin** (const simd< _Tp, _Abi > &__x, const typename _Arg2::type &__y)
- constexpr float **std::fmin** (float __x, float __y)
- constexpr long double **std::fmin** (long double __x, long double __y)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type **std::fmod** (_Tp __x, _Up __y)
- template<typename _Tp, typename _Abi, typename..., typename _Arg2 = _Extra_argument_type<_Tp, _Tp, _Abi>, typename _R = _↩ Math_return_type_t< decltype(std::fmod(declval<double>(), _Arg2::declval())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::fmod** (const simd< _Tp, _Abi > &__x, const typename _Arg2::type &__y)
- constexpr float **std::fmod** (float __x, float __y)
- constexpr long double **std::fmod** (long double __x, long double __y)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, int >::__type **std::fpclassify** (_Tp __x)
- constexpr int **std::fpclassify** (double __x)
- constexpr int **std::fpclassify** (float __x)
- constexpr int **std::fpclassify** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::frexp** (_Tp __x, int *__exp)

- template<typename _Tp, typename _Abi, typename = __detail::__odr_helper>
  enable_if_t< is_floating_point_v< _Tp >, simd< _Tp, _Abi > > std::frexp (const simd< _Tp, _Abi > &__x, _Samesize< int, simd< _Tp, _Abi > > *__exp)
- float **std::frexp** (float __x, int *__exp)
- long double **std::frexp** (long double __x, int *__exp)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type **std::hypot** (_Tp __x, _Up __y)
- template<typename _Tp, typename _Abi>
  _GLIBCXX_SIMD_INTRINSIC simd< _Tp, _Abi > **std::hypot** (const simd< _Tp, _Abi > &__x, const simd< _Tp, _Abi > &__y)
- constexpr float **std::hypot** (float __x, float __y)
- constexpr long double **std::hypot** (long double __x, long double __y)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, int >::__type **std::ilogb** (_Tp __x)
- constexpr int **std::ilogb** (float __x)
- constexpr int **std::ilogb** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::ilogb(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::ilogb** (simd< _Tp, _Abi > __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, bool >::__type **std::isfinite** (_Tp)
- constexpr bool **std::isfinite** (double __x)
- constexpr bool **std::isfinite** (float __x)
- constexpr bool **std::isfinite** (long double __x)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__enable_if<(__is_arithmetic< _Tp >::__value &&__is_arithmetic< _Up >::__value), bool >::__type **std::isgreater** (_Tp __x, _Up __y)
- constexpr bool **std::isgreater** (double __x, double __y)
- constexpr bool **std::isgreater** (float __x, float __y)
- constexpr bool **std::isgreater** (long double __x, long double __y)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__enable_if<(__is_arithmetic< _Tp >::__value &&__is_arithmetic< _Up >::__value), bool >::__type **std::isgreaterequal** (_Tp __x, _Up __y)
- constexpr bool **std::isgreaterequal** (double __x, double __y)
- constexpr bool **std::isgreaterequal** (float __x, float __y)
- constexpr bool **std::isgreaterequal** (long double __x, long double __y)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, bool >::__type **std::isinf** (_Tp)
- constexpr bool **std::isinf** (double __x)
- constexpr bool **std::isinf** (float __x)
- constexpr bool **std::isinf** (long double __x)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__enable_if<(__is_arithmetic< _Tp >::__value &&__is_arithmetic< _Up >::__value), bool >::__type **std::isless** (_Tp __x, _Up __y)
- constexpr bool **std::isless** (double __x, double __y)
- constexpr bool **std::isless** (float __x, float __y)
- constexpr bool **std::isless** (long double __x, long double __y)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__enable_if<(__is_arithmetic< _Tp >::__value &&__is_arithmetic< _Up >::__value), bool >::__type **std::islessequal** (_Tp __x, _Up __y)
- constexpr bool **std::islessequal** (double __x, double __y)

- constexpr bool **std::islessequal** (float __x, float __y)
- constexpr bool **std::islessequal** (long double __x, long double __y)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__enable_if<(__is_arithmetic< _Tp >::__value &&__is_arithmetic< _Up >::__value), bool >::__type **std::islessgreater** (_Tp __x, _Up __y)
- constexpr bool **std::islessgreater** (double __x, double __y)
- constexpr bool **std::islessgreater** (float __x, float __y)
- constexpr bool **std::islessgreater** (long double __x, long double __y)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, bool >::__type **std::isnan** (_Tp)
- constexpr bool **std::isnan** (double __x)
- constexpr bool **std::isnan** (float __x)
- constexpr bool **std::isnan** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, bool >::__type **std::isnormal** (_Tp __x)
- constexpr bool **std::isnormal** (double __x)
- constexpr bool **std::isnormal** (float __x)
- constexpr bool **std::isnormal** (long double __x)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__enable_if<(__is_arithmetic< _Tp >::__value &&__is_arithmetic< _Up >::__value), bool >::__type **std::isunordered** (_Tp __x, _Up __y)
- constexpr bool **std::isunordered** (double __x, double __y)
- constexpr bool **std::isunordered** (float __x, float __y)
- constexpr bool **std::isunordered** (long double __x, long double __y)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::ldexp** (_Tp __x, int __exp)
- template<typename _Tp, typename _Abi, typename..., typename _Arg2 = _Extra_argument_type<int, _Tp, _Abi>, typename _R = _Math↩
  _return_type_t< decltype(std::ldexp(declval<double>(), _Arg2::declval())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::ldexp** (const simd< _Tp, _Abi > &__x, const typename _Arg2::type &__y)
- constexpr float **std::ldexp** (float __x, int __exp)
- constexpr long double **std::ldexp** (long double __x, int __exp)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::lgamma** (_Tp __x)
- constexpr float **std::lgamma** (float __x)
- constexpr long double **std::lgamma** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::lgamma(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::lgamma** (simd< _Tp, _Abi > __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, longlong >::__type **std::llrint** (_Tp __x)
- constexpr long long **std::llrint** (float __x)
- constexpr long long **std::llrint** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::llrint(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::llrint** (simd< _Tp, ↩
  _Abi > __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, longlong >::__type **std::llround** (_Tp __x)
- constexpr long long **std::llround** (float __x)

- constexpr long long **std::llround** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::llround(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::llround** (simd< _Tp, _Abi > __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::log** (_Tp __x)
- constexpr float **std::log** (float __x)
- constexpr long double **std::log** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::log(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::log** (simd< _Tp, _Abi > __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::log10** (_Tp __x)
- constexpr float **std::log10** (float __x)
- constexpr long double **std::log10** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::log10(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::log10** (simd< _Tp, _Abi > __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::log1p** (_Tp __x)
- constexpr float **std::log1p** (float __x)
- constexpr long double **std::log1p** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::log1p(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::log1p** (simd< _Tp, _Abi > __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::log2** (_Tp __x)
- constexpr float **std::log2** (float __x)
- constexpr long double **std::log2** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::log2(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::log2** (simd< _Tp, _Abi > __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::logb** (_Tp __x)
- template<typename _Tp, typename _Abi, typename = __detail::__odr_helper>
  enable_if_t< is_floating_point< _Tp >::value, simd< _Tp, _Abi > > **std::logb** (const simd< _Tp, _Abi > &__x)
- constexpr float **std::logb** (float __x)
- constexpr long double **std::logb** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, long >::__type **std::lrint** (_Tp __x)
- constexpr long **std::lrint** (float __x)
- constexpr long **std::lrint** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::lrint(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::lrint** (simd< _Tp, _Abi > __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, long >::__type **std::lround** (_Tp __x)

- constexpr long **std::lround** (float __x)
- constexpr long **std::lround** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::lround(declval<double>())), ↩ _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::lround** (simd< _Tp, _Abi > __x)
- template<typename _Tp, typename _Abi, typename = __detail::__odr_helper>
  enable_if_t< is_floating_point_v< _Tp >, simd< _Tp, _Abi > > **std::modf** (const simd< _Tp, _Abi > &__x, simd< _Tp, _Abi > *__iptr)
- float **std::modf** (float __x, float *__iptr)
- long double **std::modf** (long double __x, long double *__iptr)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::nearbyint** (_Tp __x)
- constexpr float **std::nearbyint** (float __x)
- constexpr long double **std::nearbyint** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::nearbyint(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::nearbyint** (simd< ↩ _Tp, _Abi > __x)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type **std::nextafter** (_Tp __x, _Up __y)
- template<typename _Tp, typename _Abi, typename..., typename _Arg2 = _Extra_argument_type<_Tp, _Tp, _Abi>, typename _R = _↩ Math_return_type_t< decltype(std::nextafter(declval<double>(), _Arg2::declval())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::nextafter** (const simd< _Tp, _Abi > &__x, const typename _Arg2::type &__y)
- constexpr float **std::nextafter** (float __x, float __y)
- constexpr long double **std::nextafter** (long double __x, long double __y)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::nexttoward** (_Tp ↩ __x, long double __y)
- constexpr float **std::nexttoward** (float __x, long double __y)
- constexpr long double **std::nexttoward** (long double __x, long double __y)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type **std::pow** (_Tp __x, _Up __y)
- template<typename _Tp, typename _Abi, typename..., typename _Arg2 = _Extra_argument_type<_Tp, _Tp, _Abi>, typename _R = _↩ Math_return_type_t< decltype(std::pow(declval<double>(), _Arg2::declval())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::pow** (const simd< _Tp, _Abi > &__x, const typename _Arg2::type &__y)
- constexpr float **std::pow** (float __x, float __y)
- constexpr long double **std::pow** (long double __x, long double __y)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type **std::remainder** (_Tp __x, _Up __y)
- template<typename _Tp, typename _Abi, typename..., typename _Arg2 = _Extra_argument_type<_Tp, _Tp, _Abi>, typename _R = _↩ Math_return_type_t< decltype(std::remainder(declval<double>(), _Arg2::declval())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::remainder** (const simd< _Tp, _Abi > &__x, const typename _Arg2::type &__y)
- constexpr float **std::remainder** (float __x, float __y)
- constexpr long double **std::remainder** (long double __x, long double __y)
- template<typename _Tp, typename _Up>
  __gnu_cxx::__promote_2< _Tp, _Up >::__type **std::remquo** (_Tp __x, _Up __y, int *__pquo)

- template<typename _Tp, typename _Abi, typename..., typename _Arg2 = _Extra_argument_type<_Tp, _Tp, _Abi>, typename _Arg3 = _Extra_argument_type<int*, _Tp, _Abi>, typename _R = _Math_return_type_t< decltype(std::remquo(declval<double>(), _Arg2::declval(), _Arg3::declval())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::remquo** (const simd< _Tp, _Abi > &__x, const typename _Arg2::type &__y, const typename _Arg3::type &__z)
- float **std::remquo** (float __x, float __y, int *__pquo)
- long double **std::remquo** (long double __x, long double __y, int *__pquo)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::rint** (_Tp __x)
- constexpr float **std::rint** (float __x)
- constexpr long double **std::rint** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::rint(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::rint** (simd< _Tp, _Abi > __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::round** (_Tp __x)
- constexpr float **std::round** (float __x)
- constexpr long double **std::round** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::round(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::round** (simd< _Tp, _Abi > __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::scalbln** (_Tp __x, long __ex)
- template<typename _Tp, typename _Abi, typename..., typename _Arg2 = _Extra_argument_type<long, _Tp, _Abi>, typename _R = _Math_return_type_t< decltype(std::scalbln(declval<double>(), _Arg2::declval())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::scalbln** (const simd< _Tp, _Abi > &__x, const typename _Arg2::type &__y)
- constexpr float **std::scalbln** (float __x, long __ex)
- constexpr long double **std::scalbln** (long double __x, long __ex)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::scalbn** (_Tp __x, int __ex)
- template<typename _Tp, typename _Abi, typename..., typename _Arg2 = _Extra_argument_type<int, _Tp, _Abi>, typename _R = _Math_return_type_t< decltype(std::scalbn(declval<double>(), _Arg2::declval())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::scalbn** (const simd< _Tp, _Abi > &__x, const typename _Arg2::type &__y)
- constexpr float **std::scalbn** (float __x, int __ex)
- constexpr long double **std::scalbn** (long double __x, int __ex)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, bool >::__type **std::signbit** (_Tp __x)
- constexpr bool **std::signbit** (double __x)
- constexpr bool **std::signbit** (float __x)
- constexpr bool **std::signbit** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::sin** (_Tp __x)
- template<typename _Tp, typename _Abi, typename = __detail::__odr_helper>
  enable_if_t< is_floating_point_v< _Tp >, simd< _Tp, _Abi > > **std::sin** (const simd< _Tp, _Abi > &__x)
- constexpr float **std::sin** (float __x)
- constexpr long double **std::sin** (long double __x)

- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::sinh** (_Tp __x)
- constexpr float **std::sinh** (float __x)
- constexpr long double **std::sinh** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::sinh(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::sinh** (simd< _Tp, ↩ _Abi > __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::sqrt** (_Tp __x)
- constexpr float **std::sqrt** (float __x)
- constexpr long double **std::sqrt** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::sqrt(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::sqrt** (simd< _Tp, _Abi > __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::tan** (_Tp __x)
- constexpr float **std::tan** (float __x)
- constexpr long double **std::tan** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::tan(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::tan** (simd< _Tp, _Abi > __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::tanh** (_Tp __x)
- constexpr float **std::tanh** (float __x)
- constexpr long double **std::tanh** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::tanh(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::tanh** (simd< _Tp, ↩ _Abi > __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::tgamma** (_Tp __x)
- constexpr float **std::tgamma** (float __x)
- constexpr long double **std::tgamma** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::tgamma(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::tgamma** (simd< _Tp, _Abi > __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::trunc** (_Tp __x)
- constexpr float **std::trunc** (float __x)
- constexpr long double **std::trunc** (long double __x)
- template<typename _Tp, typename _Abi, typename..., typename _R = _Math_return_type_t< decltype(std::trunc(declval<double>())), _Tp, _Abi>>
  _GLIBCXX_SIMD_ALWAYS_INLINE enable_if_t< is_floating_point_v< _Tp >, _R > **std::trunc** (simd< _Tp, _Abi > __x)

6.218.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the *.h implementation files.
This is the C++ version of the Standard C Library header `math.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

6.219 ext/cmath File Reference

Namespaces

- namespace __gnu_cxx

Macros

- #define **_EXT_CMATH**

Variables

- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__e**
- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__gamma_e**
- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__ln_10**
- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__ln_2**
- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__ln_3**
- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__log10_e**
- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__log2_e**
- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__one_div_e**
- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__one_div_pi**
- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__one_div_root_2**
- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__phi**
- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__pi**
- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__pi_half**
- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__pi_quarter**
- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__pi_third**
- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__root_2**
- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__root_3**

- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__root_5**
- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__root_7**
- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__root_pi_div_2**
- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__two_div_pi**
- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__two_div_root_pi**

6.219.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

6.220 tr1/cmath File Reference

Namespaces

- namespace std
- namespace std::tr1

Macros

- #define **_GLIBCXX_TR1_CMATH**

Functions

- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::assoc_laguerre (unsigned int __n, unsigned int __m, _Tp __x)
- float **std::tr1::assoc_laguerref** (unsigned int __n, unsigned int __m, float __x)
- long double **std::tr1::assoc_laguerrel** (unsigned int __n, unsigned int __m, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::assoc_legendre (unsigned int __l, unsigned int __m, _Tp __x)
- float **std::tr1::assoc_legendref** (unsigned int __l, unsigned int __m, float __x)
- long double **std::tr1::assoc_legendrel** (unsigned int __l, unsigned int __m, long double __x)
- template<typename _Tpx, typename _Tpy>
  __gnu_cxx::__promote_2< _Tpx, _Tpy >::__type std::tr1::beta (_Tpx __x, _Tpy __y)
- float **std::tr1::betaf** (float __x, float __y)
- long double **std::tr1::betal** (long double __x, long double __y)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::comp_ellint_1 (_Tp __k)
- float **std::tr1::comp_ellint_1f** (float __k)
- long double **std::tr1::comp_ellint_1l** (long double __k)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::comp_ellint_2 (_Tp __k)
- float **std::tr1::comp_ellint_2f** (float __k)
- long double **std::tr1::comp_ellint_2l** (long double __k)
- template<typename _Tp, typename _Tpn>
  __gnu_cxx::__promote_2< _Tp, _Tpn >::__type std::tr1::comp_ellint_3 (_Tp __k, _Tpn __nu)
- float **std::tr1::comp_ellint_3f** (float __k, float __nu)
- long double **std::tr1::comp_ellint_3l** (long double __k, long double __nu)
- template<typename _Tpa, typename _Tpc, typename _Tp>
  __gnu_cxx::__promote_3< _Tpa, _Tpc, _Tp >::__type std::tr1::conf_hyperg (_Tpa __a, _Tpc __c, _Tp __x)

- float std::tr1::conf_hypergf (float __a, float __c, float __x)
- long double std::tr1::conf_hypergl (long double __a, long double __c, long double __x)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_i (_Tpnu __nu, _Tp __x)
- float **std::tr1::cyl_bessel_if** (float __nu, float __x)
- long double **std::tr1::cyl_bessel_il** (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_j (_Tpnu __nu, _Tp __x)
- float **std::tr1::cyl_bessel_jf** (float __nu, float __x)
- long double **std::tr1::cyl_bessel_jl** (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_k (_Tpnu __nu, _Tp __x)
- float **std::tr1::cyl_bessel_kf** (float __nu, float __x)
- long double **std::tr1::cyl_bessel_kl** (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_neumann (_Tpnu __nu, _Tp __x)
- float **std::tr1::cyl_neumannf** (float __nu, float __x)
- long double **std::tr1::cyl_neumannl** (long double __nu, long double __x)
- template<typename _Tp, typename _Tpp>
  __gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::tr1::ellint_1 (_Tp __k, _Tpp __phi)
- float **std::tr1::ellint_1f** (float __k, float __phi)
- long double **std::tr1::ellint_1l** (long double __k, long double __phi)
- template<typename _Tp, typename _Tpp>
  __gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::tr1::ellint_2 (_Tp __k, _Tpp __phi)
- float **std::tr1::ellint_2f** (float __k, float __phi)
- long double **std::tr1::ellint_2l** (long double __k, long double __phi)
- template<typename _Tp, typename _Tpn, typename _Tpp>
  __gnu_cxx::__promote_3< _Tp, _Tpn, _Tpp >::__type std::tr1::ellint_3 (_Tp __k, _Tpn __nu, _Tpp __phi)
- float **std::tr1::ellint_3f** (float __k, float __nu, float __phi)
- long double **std::tr1::ellint_3l** (long double __k, long double __nu, long double __phi)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::expint (_Tp __x)
- float **std::tr1::expintf** (float __x)
- long double **std::tr1::expintl** (long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type **std::tr1::fabs** (_Tp __x)
- float **std::tr1::fabs** (float __x)
- long double **std::tr1::fabs** (long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::hermite (unsigned int __n, _Tp __x)
- float **std::tr1::hermitef** (unsigned int __n, float __x)
- long double **std::tr1::hermitel** (unsigned int __n, long double __x)
- template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp>
  __gnu_cxx::__promote_4< _Tpa, _Tpb, _Tpc, _Tp >::__type std::tr1::hyperg (_Tpa __a, _Tpb __b, _Tpc __c, _Tp __x)
- float std::tr1::hypergf (float __a, float __b, float __c, float __x)
- long double std::tr1::hypergl (long double __a, long double __b, long double __c, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::laguerre (unsigned int __n, _Tp __x)
- float **std::tr1::laguerref** (unsigned int __n, float __x)
- long double **std::tr1::laguerrel** (unsigned int __n, long double __x)

- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::legendre (unsigned int __n, _Tp __x)
- float **std::tr1::legendref** (unsigned int __n, float __x)
- long double **std::tr1::legendrel** (unsigned int __n, long double __x)
- template<typename _Tp, typename _Up>
  __gnu_cxx::__promote_2< _Tp, _Up >::__type **std::tr1::pow** (_Tp __x, _Up __y)
- float **std::tr1::pow** (float __x, float __y)
- long double **std::tr1::pow** (long double __x, long double __y)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::riemann_zeta (_Tp __x)
- float **std::tr1::riemann_zetaf** (float __x)
- long double **std::tr1::riemann_zetal** (long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::sph_bessel (unsigned int __n, _Tp __x)
- float **std::tr1::sph_besself** (unsigned int __n, float __x)
- long double **std::tr1::sph_bessell** (unsigned int __n, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::sph_legendre (unsigned int __l, unsigned int __m, _Tp __theta)
- float **std::tr1::sph_legendref** (unsigned int __l, unsigned int __m, float __theta)
- long double **std::tr1::sph_legendrel** (unsigned int __l, unsigned int __m, long double __theta)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::sph_neumann (unsigned int __n, _Tp __x)
- float **std::tr1::sph_neumannf** (unsigned int __n, float __x)
- long double **std::tr1::sph_neumannl** (unsigned int __n, long double __x)

6.220.1 Detailed Description

This is a TR1 C++ Library header.

6.221 codecvt File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CODECVT**
- #define **_GLIBCXX_CODECVT_SPECIALIZATION**(_NAME, _ELEM)
- #define **_GLIBCXX_CODECVT_SPECIALIZATION2**(_NAME, _ELEM)

Enumerations

- enum **codecvt_mode** { **consume_header** , **generate_header** , **little_endian** }

6.221.1 Detailed Description

This is a Standard C++ Library header.

6.222 complex File Reference

Classes

- class std::complex< _Tp >
- class std::complex< double >
- class std::complex< float >
- class std::complex< long double >

Namespaces

- namespace std

Macros

- #define **__glibcxx_want_complex_udls**
- #define **__glibcxx_want_constexpr_complex**
- #define **__glibcxx_want_tuple_like**
- #define **_GLIBCXX26_DECLARE_COMPLEX_TUPLE_HELPER_ACCESSOR**
- #define **_GLIBCXX_COMPLEX**

Functions

- template<typename _Tp>
  _Tp **std::__complex_abs** (const complex< _Tp > &__z)
- __complex__ double **std::__complex_acos** (__complex__ double __z)
- __complex__ float **std::__complex_acos** (__complex__ float __z)
- __complex__ long double **std::__complex_acos** (const __complex__ long double &__z)
- template<typename _Tp>
  std::complex< _Tp > **std::__complex_acos** (const std::complex< _Tp > &__z)
- __complex__ double **std::__complex_acosh** (__complex__ double __z)
- __complex__ float **std::__complex_acosh** (__complex__ float __z)
- __complex__ long double **std::__complex_acosh** (const __complex__ long double &__z)
- template<typename _Tp>
  std::complex< _Tp > **std::__complex_acosh** (const std::complex< _Tp > &__z)
- template<typename _Tp>
  _Tp **std::__complex_arg** (const complex< _Tp > &__z)
- __complex__ double **std::__complex_asin** (__complex__ double __z)
- __complex__ float **std::__complex_asin** (__complex__ float __z)
- __complex__ long double **std::__complex_asin** (const __complex__ long double &__z)
- template<typename _Tp>
  std::complex< _Tp > **std::__complex_asin** (const std::complex< _Tp > &__z)
- __complex__ double **std::__complex_asinh** (__complex__ double __z)
- __complex__ float **std::__complex_asinh** (__complex__ float __z)
- __complex__ long double **std::__complex_asinh** (const __complex__ long double &__z)
- template<typename _Tp>
  std::complex< _Tp > **std::__complex_asinh** (const std::complex< _Tp > &__z)
- __complex__ double **std::__complex_atan** (__complex__ double __z)
- __complex__ float **std::__complex_atan** (__complex__ float __z)
- __complex__ long double **std::__complex_atan** (const __complex__ long double &__z)
- template<typename _Tp>
  std::complex< _Tp > **std::__complex_atan** (const std::complex< _Tp > &__z)
- __complex__ double **std::__complex_atanh** (__complex__ double __z)

- __complex__ float **std::__complex_atanh** (__complex__ float __z)
- __complex__ long double **std::__complex_atanh** (const __complex__ long double &__z)
- template<typename _Tp>
  std::complex< _Tp > **std::__complex_atanh** (const std::complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **std::__complex_cos** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **std::__complex_cosh** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **std::__complex_exp** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **std::__complex_log** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **std::__complex_pow** (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp>
  complex< _Tp > **std::__complex_pow_unsigned** (complex< _Tp > __x, unsigned __n)
- complex< double > **std::__complex_proj** (const complex< double > &__z)
- complex< float > **std::__complex_proj** (const complex< float > &__z)
- complex< long double > **std::__complex_proj** (const complex< long double > &__z)
- template<typename _Tp>
  std::complex< _Tp > **std::__complex_proj** (const std::complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **std::__complex_sin** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **std::__complex_sinh** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **std::__complex_sqrt** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **std::__complex_tan** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **std::__complex_tanh** (const complex< _Tp > &__z)
- template<typename _Tp>
  _Tp std::abs (const complex< _Tp > &)
- template<typename _Tp>
  std::complex< _Tp > **std::acos** (const std::complex< _Tp > &)
- template<typename _Tp>
  std::complex< _Tp > **std::acosh** (const std::complex< _Tp > &)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type **std::arg** (_Tp __x)
- template<typename _Tp>
  _Tp std::arg (const complex< _Tp > &)
- template<typename _Tp>
  std::complex< _Tp > **std::asin** (const std::complex< _Tp > &)
- template<typename _Tp>
  std::complex< _Tp > **std::asinh** (const std::complex< _Tp > &)
- template<typename _Tp>
  std::complex< _Tp > **std::atan** (const std::complex< _Tp > &)
- template<typename _Tp>
  std::complex< _Tp > **std::atanh** (const std::complex< _Tp > &)
- template<typename _Tp>
  constexpr std::complex< typename __gnu_cxx::__promote< _Tp >::__type > **std::conj** (_Tp __x)

- template<typename _Tp>
  constexpr complex< _Tp > std::conj (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > std::cos (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > std::cosh (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > std::exp (const complex< _Tp > &)
- template<typename _Tp>
  _Tp std::fabs (const std::complex< _Tp > &__z)
- template<typename _Tp>
  constexpr __gnu_cxx::__promote< _Tp >::__type **std::imag** (_Tp)
- template<typename _Tp>
  constexpr _Tp **std::imag** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > std::log (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > std::log10 (const complex< _Tp > &)
- template<typename _Tp>
  constexpr __gnu_cxx::__promote< _Tp >::__type **std::norm** (_Tp __x)
- template<typename _Tp>
  _Tp constexpr std::norm (const complex< _Tp > &)
- template<typename _Tp>
  constexpr _Tp std::norm (const complex< _Tp > &__z)
- template<typename _Tp>
  constexpr complex< _Tp > std::operator+ (const complex< _Tp > &__x)
- template<typename _Tp>
  constexpr complex< _Tp > std::operator- (const complex< _Tp > &__x)
- template<typename _Tp, typename _CharT, class _Traits>
  basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, const complex< _Tp > &__x)
- template ostream & **std::operator**<< (ostream &, const complex< double > &)
- template ostream & **std::operator**<< (ostream &, const complex< float > &)
- template ostream & **std::operator**<< (ostream &, const complex< long double > &)
- template wostream & **std::operator**<< (wostream &, const complex< double > &)
- template wostream & **std::operator**<< (wostream &, const complex< float > &)
- template wostream & **std::operator**<< (wostream &, const complex< long double > &)
- template<typename _Tp, typename _CharT, class _Traits>
  basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, complex< _Tp > &__x)
- template istream & **std::operator**>> (istream &, complex< double > &)
- template istream & **std::operator**>> (istream &, complex< float > &)
- template istream & **std::operator**>> (istream &, complex< long double > &)
- template wistream & **std::operator**>> (wistream &, complex< double > &)
- template wistream & **std::operator**>> (wistream &, complex< float > &)
- template wistream & **std::operator**>> (wistream &, complex< long double > &)
- template<typename _Tp>
  complex< _Tp > std::polar (const _Tp &, const _Tp &=_Tp(0))
- template<typename _Tp>
  complex< _Tp > std::pow (const _Tp &, const complex< _Tp > &)
- template<typename _Tp, typename _Up>
  std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > **std::pow** (const _Tp &__x, const std::complex< _Up > &__y)

- template<typename _Tp>
  complex< _Tp > std::pow (const complex< _Tp > &, const _Tp &)
- template<typename _Tp>
  complex< _Tp > std::pow (const complex< _Tp > &, const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > std::pow (const complex< _Tp > &, int)
- template<typename _Tp, typename _Up>
  std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > **std::pow** (const std::complex< _Tp > &__x, const _Up &__y)
- template<typename _Tp, typename _Up>
  std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > **std::pow** (const std::complex< _Tp > &__x, const std::complex< _Up > &__y)
- template<typename _Tp>
  std::complex< typename __gnu_cxx::__promote< _Tp >::__type > **std::proj** (_Tp __x)
- template<typename _Tp>
  std::complex< _Tp > **std::proj** (const std::complex< _Tp > &)
- template<typename _Tp>
  constexpr __gnu_cxx::__promote< _Tp >::__type **std::real** (_Tp __x)
- template<typename _Tp>
  constexpr _Tp **std::real** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > std::sin (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > std::sinh (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > std::sqrt (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > std::tan (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > std::tanh (const complex< _Tp > &)

- template<typename _Tp>
  constexpr complex< _Tp > std::operator+ (const _Tp &__x, const complex< _Tp > &__y)
- template<typename _Tp>
  constexpr complex< _Tp > std::operator+ (const complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp>
  constexpr complex< _Tp > std::operator+ (const complex< _Tp > &__x, const complex< _Tp > &__y)

- template<typename _Tp>
  constexpr complex< _Tp > std::operator- (const _Tp &__x, const complex< _Tp > &__y)
- template<typename _Tp>
  constexpr complex< _Tp > std::operator- (const complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp>
  constexpr complex< _Tp > std::operator- (const complex< _Tp > &__x, const complex< _Tp > &__y)

- template<typename _Tp>
  constexpr complex< _Tp > std::operator* (const _Tp &__x, const complex< _Tp > &__y)
- template<typename _Tp>
  constexpr complex< _Tp > std::operator* (const complex< _Tp > &__x, const _Tp &__y)

- template<typename _Tp>
  constexpr complex< _Tp > std::operator* (const complex< _Tp > &__x, const complex< _Tp > &__y)

- template<typename _Tp>
  constexpr complex< _Tp > std::operator/ (const _Tp &__x, const complex< _Tp > &__y)
- template<typename _Tp>
  constexpr complex< _Tp > std::operator/ (const complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp>
  constexpr complex< _Tp > std::operator/ (const complex< _Tp > &__x, const complex< _Tp > &__y)

- template<typename _Tp>
  constexpr bool std::operator== (const complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp>
  constexpr bool std::operator== (const complex< _Tp > &__x, const complex< _Tp > &__y)

6.222.1 Detailed Description

This is a Standard C++ Library header.

6.223 tr1/complex File Reference

Namespaces

- namespace std
- namespace std::tr1

Macros

- #define **_GLIBCXX_TR1_COMPLEX**

Functions

- template<typename _Tp>
  std::complex< _Tp > **std::tr1::acos** (const std::complex< _Tp > &)
- template<typename _Tp>
  std::complex< _Tp > **std::tr1::acosh** (const std::complex< _Tp > &)
- template<typename _Tp>
  _Tp std::tr1::arg (const complex< _Tp > &)
- template<typename _Tp>
  std::complex< _Tp > **std::tr1::asin** (const std::complex< _Tp > &)
- template<typename _Tp>
  std::complex< _Tp > **std::tr1::asinh** (const std::complex< _Tp > &)
- template<typename _Tp>
  std::complex< _Tp > **std::tr1::atan** (const std::complex< _Tp > &)
- template<typename _Tp>
  std::complex< _Tp > **std::tr1::atanh** (const std::complex< _Tp > &)
- template<typename _Tp>
  std::complex< typename __gnu_cxx::__promote< _Tp >::__type > **std::tr1::conj** (_Tp __x)
- template<typename _Tp>
  std::complex< _Tp > **std::tr1::conj** (const std::complex< _Tp > &__z)

- template<typename _Tp>
  std::complex< _Tp > std::tr1::fabs (const std::complex< _Tp > &__z)
- template<typename _Tp>
  constexpr _Tp **std::tr1::imag** (const complex< _Tp > &__z)
- template<typename _Tp>
  _Tp constexpr std::tr1::norm (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > std::tr1::polar (const _Tp &, const _Tp &=_Tp(0))
- template<typename _Tp, typename _Up>
  std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > **std::tr1::polar** (const _Tp &__rho, const _Up &__theta)
- template<typename _Tp>
  std::complex< _Tp > **std::tr1::pow** (const _Tp &__x, const std::complex< _Tp > &__y)
- template<typename _Tp, typename _Up>
  std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > **std::tr1::pow** (const _Tp &__x, const std::complex< _Up > &__y)
- template<typename _Tp>
  std::complex< _Tp > **std::tr1::pow** (const std::complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp, typename _Up>
  std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::tr1::pow (const std::complex< _Tp > &__x, const _Up &__y)
- template<typename _Tp>
  std::complex< _Tp > **std::tr1::pow** (const std::complex< _Tp > &__x, const std::complex< _Tp > &__y)
- template<typename _Tp, typename _Up>
  std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > **std::tr1::pow** (const std::complex< _Tp > &__x, const std::complex< _Up > &__y)
- template<typename _Tp>
  constexpr _Tp **std::tr1::real** (const complex< _Tp > &__z)

6.223.1 Detailed Description

This is a TR1 C++ Library header.

6.224 complex.h File Reference

6.224.1 Detailed Description

This is a Standard C++ Library header.

6.225 concepts File Reference

Macros

- #define **__glibcxx_want_concepts**
- #define **_GLIBCXX_CONCEPTS**

6.225.1 Detailed Description

This is a Standard C++ Library header.

6.226 condition_variable File Reference

Classes

- class std::condition_variable
- class std::condition_variable_any

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CONDITION_VARIABLE**

Enumerations

- enum class std::cv_status { **no_timeout** , **timeout** }

Functions

- void **std::notify_all_at_thread_exit** (condition_variable &, unique_lock< mutex >)

6.226.1 Detailed Description

This is a Standard C++ Library header.

6.227 csetjmp File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CSETJMP**
- #define **setjmp**(env)

6.227.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the *.h implementation files.
This is the C++ version of the Standard C Library header `setjmp.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

6.228 csignal File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CSIGNAL**

6.228.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the *.h implementation files.
This is the C++ version of the Standard C Library header `signal.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

6.229 cstdalign File Reference

Macros

- #define **_GLIBCXX_CSTDALIGN**

6.229.1 Detailed Description

This is a Standard C++ Library header.

Since

C++11 (removed in C++20)

6.230 cstdarg File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CSTDARG**
- #define **va_end**(ap)

6.230.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the *.h implementation files.
This is the C++ version of the Standard C Library header `stdarg.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

6.231 tr1/cstdarg File Reference

Macros

- #define **_GLIBCXX_TR1_CSTDARG**

6.231.1 Detailed Description

This is a TR1 C++ Library header.

6.232 cstdbool File Reference

Macros

- #define **_GLIBCXX_CSTDBOOL**

6.232.1 Detailed Description

This is a Standard C++ Library header.

Since

C++11 (removed in C++20)

6.233 tr1/cstdbool File Reference

Macros

- #define **_GLIBCXX_TR1_CSTDBOOL**

6.233.1 Detailed Description

This is a TR1 C++ Library header.

6.234 cstddef File Reference

Namespaces

- namespace std

Macros

- #define **__glibcxx_want_byte**
- #define **_GLIBCXX_CSTDDEF**

6.234.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*`.h implementation files.
This is the C++ version of the Standard C Library header `stddef.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

6.235 cstdint File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CSTDINT**

6.235.1 Detailed Description

This is a Standard C++ Library header.

6.236 tr1/cstdint File Reference

Namespaces

- namespace std
- namespace std::tr1

Macros

- #define **_GLIBCXX_TR1_CSTDINT**

6.236.1 Detailed Description

This is a TR1 C++ Library header.

6.237 cstdio File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CSTDIO**

6.237.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*`.h implementation files.
This is the C++ version of the Standard C Library header `stdio.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

6.238 tr1/cstdio File Reference

Macros

- #define **_GLIBCXX_TR1_CSTDIO**

6.238.1 Detailed Description

This is a TR1 C++ Library header.

6.239 cstdlib File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CSTDLIB**
- #define **_GLIBCXX_INCLUDE_NEXT_C_HEADERS**

Functions

- ldiv_t **std::div** (long __i, long __j) noexcept

6.239.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*`.h implementation files.
This is the C++ version of the Standard C Library header `stdlib.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

6.240 tr1/cstdlib File Reference

Macros

- #define **_GLIBCXX_TR1_CSTDLIB**

6.240.1 Detailed Description

This is a TR1 C++ Library header.

6.241 cstring File Reference

Namespaces

- namespace std

Macros

- #define **__glibcxx_want_freestanding_cstring**
- #define **_GLIBCXX_CSTRING**

Functions

- void * **std::memchr** (void *__s, int __c, size_t __n)
- char * **std::strchr** (char *__s, int __n)
- char * **std::strpbrk** (char *__s1, const char *__s2)
- char * **std::strrchr** (char *__s, int __n)
- char * **std::strstr** (char *__s1, const char *__s2)

6.241.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the *.h implementation files.
This is the C++ version of the Standard C Library header `string.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

6.242 ctgmath File Reference

Macros

- #define **_GLIBCXX_CTGMATH**

6.242.1 Detailed Description

This is a Standard C++ Library header.

Since

C++11 (removed in C++20)

6.243 tr1/ctgmath File Reference

Macros

- #define **_GLIBCXX_TR1_CTGMATH**

6.243.1 Detailed Description

This is a TR1 C++ Library header.

6.244 ctime File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CTIME**

6.244.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the ∗.h implementation files.
This is the C++ version of the Standard C Library header `time.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

6.245 tr1/ctime File Reference

Macros

- #define **_GLIBCXX_TR1_CTIME**

6.245.1 Detailed Description

This is a TR1 C++ Library header.

6.246 cuchar File Reference

Macros

- #define **_GLIBCXX_CUCHAR**

6.246.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the ∗.h implementation files.
This is the C++ version of the Standard C Library header `uchar.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

6.247 cwchar File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CWCHAR**

Functions

- wchar_t ∗ **std::wcschr** (wchar_t ∗__p, wchar_t __c)
- wchar_t ∗ **std::wcspbrk** (wchar_t ∗__s1, const wchar_t ∗__s2)
- wchar_t ∗ **std::wcsrchr** (wchar_t ∗__p, wchar_t __c)
- wchar_t ∗ **std::wcsstr** (wchar_t ∗__s1, const wchar_t ∗__s2)
- wchar_t ∗ **std::wmemchr** (wchar_t ∗__p, wchar_t __c, size_t __n)

6.247.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the *.h implementation files.
This is the C++ version of the Standard C Library header `wchar.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

6.248 tr1/cwchar File Reference

Namespaces

- namespace std
- namespace std::tr1

Macros

- #define **_GLIBCXX_TR1_CWCHAR**

6.248.1 Detailed Description

This is a TR1 C++ Library header.

6.249 cwctype File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CWCTYPE**

6.249.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the *.h implementation files.
This is the C++ version of the Standard C Library header `wctype.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

6.250 tr1/cwctype File Reference

Namespaces

- namespace std
- namespace std::tr1

Macros

- #define **_GLIBCXX_TR1_CWCTYPE**

6.250.1 Detailed Description

This is a TR1 C++ Library header.

6.251 assertions.h File Reference

Macros

- #define **__glibcxx_requires_non_empty_range**(_First, _Last)
- #define **__glibcxx_requires_nonempty**()
- #define **__glibcxx_requires_subscript**(_N)
- #define **_GLIBCXX_DEBUG_ASSERT**(_Condition)
- #define **_GLIBCXX_DEBUG_ONLY**(_Statement)
- #define **_GLIBCXX_DEBUG_PEDASSERT**(_Condition)

6.251.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

6.252 debug.h File Reference

Namespaces

- namespace __gnu_debug
- namespace std
- namespace std::__debug

Macros

- #define **__glibcxx_requires_can_decrement_range**(_First1, _Last1, _First2)
- #define **__glibcxx_requires_can_increment**(_First, _Size)
- #define **__glibcxx_requires_can_increment_range**(_First1, _Last1, _First2)
- #define **__glibcxx_requires_cond**(_Cond, _Msg)
- #define **__glibcxx_requires_heap**(_First, _Last)
- #define **__glibcxx_requires_heap_pred**(_First, _Last, _Pred)
- #define **__glibcxx_requires_irreflexive**(_First, _Last)
- #define **__glibcxx_requires_irreflexive2**(_First, _Last)
- #define **__glibcxx_requires_irreflexive_pred**(_First, _Last, _Pred)
- #define **__glibcxx_requires_irreflexive_pred2**(_First, _Last, _Pred)
- #define **__glibcxx_requires_partitioned_lower**(_First, _Last, _Value)
- #define **__glibcxx_requires_partitioned_lower_pred**(_First, _Last, _Value, _Pred)
- #define **__glibcxx_requires_partitioned_upper**(_First, _Last, _Value)
- #define **__glibcxx_requires_partitioned_upper_pred**(_First, _Last, _Value, _Pred)
- #define **__glibcxx_requires_sorted**(_First, _Last)
- #define **__glibcxx_requires_sorted_pred**(_First, _Last, _Pred)
- #define **__glibcxx_requires_sorted_set**(_First1, _Last1, _First2)
- #define **__glibcxx_requires_sorted_set_pred**(_First1, _Last1, _First2, _Pred)
- #define **__glibcxx_requires_string**(_String)
- #define **__glibcxx_requires_string_len**(_String, _Len)
- #define **__glibcxx_requires_valid_range**(_First, _Last)

6.252.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

6.253 formatter.h File Reference

Classes

- class __gnu_debug::type_info

Namespaces

- namespace __gnu_cxx
- namespace __gnu_debug
- namespace std

Macros

- #define **_GLIBCXX_TYPEID**(_Type)

Enumerations

- enum **_Debug_msg_id** {
  __msg_valid_range , **__msg_insert_singular** , **__msg_insert_different** , **__msg_erase_bad** ,
  __msg_erase_different , **__msg_subscript_oob** , **__msg_empty** , **__msg_unpartitioned** ,
  __msg_unpartitioned_pred , **__msg_unsorted** , **__msg_unsorted_pred** , **__msg_not_heap** ,
  __msg_not_heap_pred , **__msg_bad_bitset_write** , **__msg_bad_bitset_read** , **__msg_bad_bitset_flip** ,
  __msg_self_splice , **__msg_splice_alloc** , **__msg_splice_bad** , **__msg_splice_other** ,
  __msg_splice_overlap , **__msg_init_singular** , **__msg_init_copy_singular** , **__msg_init_const_singular** ,
  __msg_copy_singular , **__msg_bad_deref** , **__msg_bad_inc** , **__msg_bad_dec** ,
  __msg_iter_subscript_oob , **__msg_advance_oob** , **__msg_retreat_oob** , **__msg_iter_compare_bad** ,
  __msg_compare_different , **__msg_iter_order_bad** , **__msg_order_different** , **__msg_distance_bad** ,
  __msg_distance_different , **__msg_deref_istream** , **__msg_inc_istream** , **__msg_output_ostream** ,
  __msg_deref_istreambuf , **__msg_inc_istreambuf** , **__msg_insert_after_end** , **__msg_erase_after_bad** ,
  __msg_valid_range2 , **__msg_local_iter_compare_bad** , **__msg_non_empty_range** , **__msg_self_move↩
  _assign** ,
  __msg_bucket_index_oob , **__msg_valid_load_factor** , **__msg_equal_allocs** , **__msg_insert_range_↩
  from_self** ,
  __msg_irreflexive_ordering }

Functions

- template<typename _Iterator>
  constexpr bool **__gnu_debug::__check_singular** (_Iterator const &)

6.253.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

6.254 functions.h File Reference

Namespaces

- namespace __gnu_debug

Functions

- template<typename _ForwardIterator, typename _Tp>
  constexpr bool **__gnu_debug::__check_partitioned_lower** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)

- template<typename _ForwardIterator, typename _Tp, typename _Pred>
  constexpr bool **__gnu_debug::__check_partitioned_lower** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value, _Pred __pred)
- template<typename _ForwardIterator, typename _Tp>
  constexpr bool **__gnu_debug::__check_partitioned_upper** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)
- template<typename _ForwardIterator, typename _Tp, typename _Pred>
  constexpr bool **__gnu_debug::__check_partitioned_upper** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value, _Pred __pred)
- template<typename _InputIterator>
  constexpr bool **__gnu_debug::__check_sorted** (const _InputIterator &__first, const _InputIterator &__last)
- template<typename _InputIterator, typename _Predicate>
  constexpr bool **__gnu_debug::__check_sorted** (const _InputIterator &__first, const _InputIterator &__last, _↩ Predicate __pred)
- template<typename _ForwardIterator, typename _Predicate>
  constexpr bool **__gnu_debug::__check_sorted_aux** (_ForwardIterator __first, _ForwardIterator __last, _↩ Predicate __pred, std::forward_iterator_tag)
- template<typename _ForwardIterator>
  constexpr bool **__gnu_debug::__check_sorted_aux** (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- template<typename _InputIterator, typename _Predicate>
  constexpr bool **__gnu_debug::__check_sorted_aux** (const _InputIterator &, const _InputIterator &, _Predicate, std::input_iterator_tag)
- template<typename _InputIterator>
  constexpr bool **__gnu_debug::__check_sorted_aux** (const _InputIterator &, const _InputIterator &, std::input_iterator_tag)
- template<typename _InputIterator1, typename _InputIterator2>
  constexpr bool **__gnu_debug::__check_sorted_set** (const _InputIterator1 &__first, const _InputIterator1 &__↩ last, const _InputIterator2 &)
- template<typename _InputIterator1, typename _InputIterator2, typename _Predicate>
  constexpr bool **__gnu_debug::__check_sorted_set** (const _InputIterator1 &__first, const _InputIterator1 &__↩ last, const _InputIterator2 &, _Predicate __pred)
- template<typename _InputIterator, typename _Predicate>
  constexpr bool **__gnu_debug::__check_sorted_set_aux** (const _InputIterator &, const _InputIterator &, _↩ Predicate, std::__false_type)
- template<typename _InputIterator>
  constexpr bool **__gnu_debug::__check_sorted_set_aux** (const _InputIterator &, const _InputIterator &, std::↩ __false_type)
- template<typename _InputIterator, typename _Predicate>
  constexpr bool **__gnu_debug::__check_sorted_set_aux** (const _InputIterator &__first, const _InputIterator &↩ __last, _Predicate __pred, std::__true_type)
- template<typename _InputIterator>
  constexpr bool **__gnu_debug::__check_sorted_set_aux** (const _InputIterator &__first, const _InputIterator &↩ __last, std::__true_type)
- template<typename _InputIterator>
  _InputIterator **__gnu_debug::__check_valid_range** (const _InputIterator &__first, const _InputIterator &__last, const char *__file, unsigned int __line, const char *__function)
- template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator>
  bool **__gnu_debug::__foreign_iterator** (const _Safe_iterator< _Iterator, _Sequence, _Category > &__it, _↩ InputIterator __other, _InputIterator __other_end)
- template<typename _Iterator, typename _Sequence, typename _Category, typename _Integral>
  bool **__gnu_debug::__foreign_iterator_aux** (const _Safe_iterator< _Iterator, _Sequence, _Category > &, _↩ Integral, _Integral, std::__true_type)

- template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator>
  bool **__gnu_debug::__foreign_iterator_aux** (const _Safe_iterator< _Iterator, _Sequence, _Category > &__it, _InputIterator __other, _InputIterator __other_end, std::__false_type)
- template<typename _Iterator, typename _Sequence, typename _Category, typename _OtherIterator, typename _OtherSequence, typename _OtherCategory>
  bool __gnu_debug::__foreign_iterator_aux2 (const _Safe_iterator< _Iterator, _Sequence, _Category > &, const _Safe_iterator< _OtherIterator, _OtherSequence, _OtherCategory > &, const _Safe_iterator< _OtherIterator, _OtherSequence, _OtherCategory > &)
- template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator>
  bool **__gnu_debug::__foreign_iterator_aux2** (const _Safe_iterator< _Iterator, _Sequence, _Category > &__it, const _InputIterator &__other, const _InputIterator &__other_end)
- template<typename _Iterator, typename _Sequence, typename _Category, typename _OtherIterator>
  bool __gnu_debug::__foreign_iterator_aux2 (const _Safe_iterator< _Iterator, _Sequence, _Category > &__it, const _Safe_iterator< _OtherIterator, _Sequence, _Category > &__other, const _Safe_iterator< _OtherIterator, _Sequence, _Category > &)
- template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator>
  bool **__gnu_debug::__foreign_iterator_aux3** (const _Safe_iterator< _Iterator, _Sequence, _Category > &, const _InputIterator &, const _InputIterator &, std::__false_type)
- template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator>
  bool **__gnu_debug::__foreign_iterator_aux3** (const _Safe_iterator< _Iterator, _Sequence, _Category > &__it, const _InputIterator &__other, const _InputIterator &__other_end, std::__true_type)
- template<typename _Iterator, typename _Sequence, typename _Category>
  bool **__gnu_debug::__foreign_iterator_aux4** (const _Safe_iterator< _Iterator, _Sequence, _Category > &,...)
- template<typename _Iterator, typename _Sequence, typename _Category>
  bool **__gnu_debug::__foreign_iterator_aux4** (const _Safe_iterator< _Iterator, _Sequence, _Category > &__it, const typename _Sequence::value_type *__other)
- template<typename _Iterator>
  constexpr bool **__gnu_debug::__is_irreflexive** (_Iterator __it)
- template<typename _Iterator, typename _Pred>
  constexpr bool **__gnu_debug::__is_irreflexive_pred** (_Iterator __it, _Pred __pred)

6.254.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

6.255 helper_functions.h File Reference

Namespaces

- namespace __gnu_debug

Enumerations

- enum __gnu_debug::_Distance_precision {
  __dp_none , **__dp_equality** , **__dp_sign** , **__dp_sign_max_size** ,
  __dp_exact }

Functions

- template<typename _Iterator>
  constexpr _Iterator __gnu_debug::__base (_Iterator __it)
- template<typename _InputIterator, typename _Size>
  constexpr bool **__gnu_debug::__can_advance** (_InputIterator, _Size)

- template<typename _InputIterator, typename _Diff>
  constexpr bool **__gnu_debug::__can_advance** (_InputIterator, const std::pair< _Diff, _Distance_precision > &, int)
- template<typename _Iterator, typename _Sequence, typename _Category, typename _Size>
  bool **__gnu_debug::__can_advance** (const _Safe_iterator< _Iterator, _Sequence, _Category > &, _Size)
- template<typename _Iterator, typename _Sequence, typename _Category, typename _Diff>
  bool **__gnu_debug::__can_advance** (const _Safe_iterator< _Iterator, _Sequence, _Category > &, const std::pair< _Diff, _Distance_precision > &, int)
- template<typename _Iterator>
  constexpr bool **__gnu_debug::__check_singular** (_Iterator const &)
- template<typename _Tp>
  constexpr bool __gnu_debug::__check_singular (_Tp *const &__ptr)
- bool **__gnu_debug::__check_singular_aux** (const class _Safe_iterator_base *)
- bool **__gnu_debug::__check_singular_aux** (const void *)
- template<typename _Iterator>
  constexpr _Distance_traits< _Iterator >::__type **__gnu_debug::__get_distance** (_Iterator __lhs, _Iterator __rhs)
- template<typename _Iterator>
  constexpr _Distance_traits< _Iterator >::__type **__gnu_debug::__get_distance** (_Iterator __lhs, _Iterator __rhs, std::input_iterator_tag)
- template<typename _Iterator>
  constexpr _Distance_traits< _Iterator >::__type __gnu_debug::__get_distance (_Iterator __lhs, _Iterator __rhs, std::random_access_iterator_tag)
- template<typename _Iterator>
  constexpr _Iterator **__gnu_debug::__unsafe** (_Iterator __it)
- template<typename _InputIterator>
  constexpr bool **__gnu_debug::__valid_range** (_InputIterator __first, _InputIterator __last)
- template<typename _InputIterator>
  constexpr bool __gnu_debug::__valid_range (_InputIterator __first, _InputIterator __last, typename _Distance_traits< _InputIterator >::__type &__dist)
- template<typename _Iterator, typename _Sequence, typename _Category>
  bool **__gnu_debug::__valid_range** (const _Safe_iterator< _Iterator, _Sequence, _Category > &, const _Safe_iterator< _Iterator, _Sequence, _Category > &)
- template<typename _Iterator, typename _Sequence, typename _Category>
  bool __gnu_debug::__valid_range (const _Safe_iterator< _Iterator, _Sequence, _Category > &, const _Safe_iterator< _Iterator, _Sequence, _Category > &, typename _Distance_traits< _Iterator >::__type &)
- template<typename _Iterator, typename _Sequence>
  bool **__gnu_debug::__valid_range** (const _Safe_local_iterator< _Iterator, _Sequence > &, const _Safe_local_iterator< _Iterator, _Sequence > &)
- template<typename _Iterator, typename _Sequence>
  bool **__gnu_debug::__valid_range** (const _Safe_local_iterator< _Iterator, _Sequence > &, const _Safe_local_iterator< _Iterator, _Sequence > &, typename _Distance_traits< _Iterator >::__type &)
- template<typename _InputIterator>
  constexpr bool __gnu_debug::__valid_range_aux (_InputIterator __first, _InputIterator __last, std::__false_type)
- template<typename _InputIterator>
  constexpr bool **__gnu_debug::__valid_range_aux** (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- template<typename _InputIterator>
  constexpr bool **__gnu_debug::__valid_range_aux** (_InputIterator __first, _InputIterator __last, std::random_access_iterator_tag)
- template<typename _InputIterator>
  constexpr bool **__gnu_debug::__valid_range_aux** (_InputIterator __first, _InputIterator __last, typename _Distance_traits< _InputIterator >::__type &__dist, std::__false_type)
- template<typename _Integral>
  constexpr bool __gnu_debug::__valid_range_aux (_Integral, _Integral, std::__true_type)

- template<typename _Integral>
  constexpr bool **__gnu_debug::__valid_range_aux** (_Integral, _Integral, typename _Distance_traits< _Integral >::__type &__dist, std::__true_type)

6.255.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

6.256 macros.h File Reference

Macros

- #define **__glibcxx_check_bucket_index**(_N)
- #define **__glibcxx_check_can_decrement_range**(_First1, _Last1, _First2)
- #define **__glibcxx_check_can_increment**(_First, _Size)
- #define **__glibcxx_check_can_increment_dist**(_First, _Dist, _Way)
- #define **__glibcxx_check_can_increment_range**(_First1, _Last1, _First2)
- #define **__glibcxx_check_equal_allocs**(_This, _Other)
- #define __glibcxx_check_erase(_Position)
- #define **__glibcxx_check_erase2**(_CPosition)
- #define __glibcxx_check_erase_after(_Position)
- #define __glibcxx_check_erase_range(_First, _Last)
- #define __glibcxx_check_erase_range_after(_First, _Last)
- #define **__glibcxx_check_heap**(_First, _Last)
- #define __glibcxx_check_heap_pred(_First, _Last, _Pred)
- #define __glibcxx_check_insert(_Position)
- #define __glibcxx_check_insert_after(_Position)
- #define __glibcxx_check_insert_range(_Position, _First, _Last, _Dist)
- #define __glibcxx_check_insert_range_after(_Position, _First, _Last, _Dist)
- #define **__glibcxx_check_irreflexive**(_First, _Last)
- #define **__glibcxx_check_irreflexive2**(_First, _Last)
- #define **__glibcxx_check_irreflexive_pred**(_First, _Last, _Pred)
- #define **__glibcxx_check_irreflexive_pred2**(_First, _Last, _Pred)
- #define **__glibcxx_check_max_load_factor**(_F)
- #define **__glibcxx_check_non_empty_range**(_First, _Last)
- #define **__glibcxx_check_nonempty**()
- #define __glibcxx_check_partitioned_lower(_First, _Last, _Value)
- #define __glibcxx_check_partitioned_lower_pred(_First, _Last, _Value, _Pred)
- #define **__glibcxx_check_partitioned_upper**(_First, _Last, _Value)
- #define __glibcxx_check_partitioned_upper_pred(_First, _Last, _Value, _Pred)
- #define **__glibcxx_check_sorted**(_First, _Last)
- #define __glibcxx_check_sorted_pred(_First, _Last, _Pred)
- #define **__glibcxx_check_sorted_set**(_First1, _Last1, _First2)
- #define **__glibcxx_check_sorted_set_pred**(_First1, _Last1, _First2, _Pred)
- #define **__glibcxx_check_subscript**(_N)
- #define **__glibcxx_check_valid_constructor_range**(_First, _Last)
- #define **__glibcxx_check_valid_range**(_First, _Last)
- #define **__glibcxx_check_valid_range2**(_First, _Last, _Dist)
- #define **__glibcxx_check_valid_range_at**(_First, _Last, _File, _Line, _Func)
- #define **_GLIBCXX_DEBUG_VERIFY**(_Cond, _ErrMsg)
- #define **_GLIBCXX_DEBUG_VERIFY_AT**(_Cond, _ErrMsg, _File, _Line)
- #define _GLIBCXX_DEBUG_VERIFY_AT_F(_Cond, _ErrMsg, _File, _Line, _Func)

6.256.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

6.256.2 Macro Definition Documentation

__glibcxx_check_erase

```
#define __glibcxx_check_erase(
            _Position)
```

Verify that we can erase the element referenced by the iterator _Position. We can erase the element if the _Position iterator is dereferenceable and references this sequence.

__glibcxx_check_erase_after

```
#define __glibcxx_check_erase_after(
            _Position)
```

Verify that we can erase the element after the iterator _Position. We can erase the element if the _Position iterator is before a dereferenceable one and references this sequence.

__glibcxx_check_erase_range

```
#define __glibcxx_check_erase_range(
            _First,
            _Last)
```

Verify that we can erase the elements in the iterator range [_First, _Last). We can erase the elements if [_First, _Last) is a valid iterator range within this sequence.

__glibcxx_check_erase_range_after

```
#define __glibcxx_check_erase_range_after(
            _First,
            _Last)
```

Verify that we can erase the elements in the iterator range (_First, _Last). We can erase the elements if (_First, _Last) is a valid iterator range within this sequence.

__glibcxx_check_heap_pred

```
#define __glibcxx_check_heap_pred(
            _First,
            _Last,
            _Pred)
```

Verify that the iterator range [_First, _Last) is a heap w.r.t. the predicate _Pred.

__glibcxx_check_insert

```
#define __glibcxx_check_insert(
            _Position)
```

Verify that we can insert into ∗this with the iterator _Position. Insertion into a container at a specific position requires that the iterator be nonsingular, either dereferenceable or past-the-end, and that it reference the sequence we are inserting into. Note that this macro is only valid when the container is a_Safe_sequence and the iterator is a _Safe_iterator.

__glibcxx_check_insert_after

```
#define __glibcxx_check_insert_after(
            _Position)
```

Verify that we can insert into ∗this after the iterator _Position. Insertion into a container after a specific position requires that the iterator be nonsingular, either dereferenceable or before-begin, and that it reference the sequence we are inserting into. Note that this macro is only valid when the container is a_Safe_sequence and the iterator is a _Safe_↩ iterator.

__glibcxx_check_insert_range

```
#define __glibcxx_check_insert_range(
              _Position,
              _First,
              _Last,
              _Dist)
```

Verify that we can insert the values in the iterator range [_First, _Last) into ∗this with the iterator _Position. Insertion into a container at a specific position requires that the iterator be nonsingular (i.e., either dereferenceable or past-the-end), that it reference the sequence we are inserting into, and that the iterator range [_First, _Last) is a valid (possibly empty) range which does not reference the sequence we are inserting into. Note that this macro is only valid when the container is a _Safe_sequence and the _Position iterator is a _Safe_iterator.

__glibcxx_check_insert_range_after

```
#define __glibcxx_check_insert_range_after(
              _Position,
              _First,
              _Last,
              _Dist)
```

Verify that we can insert the values in the iterator range [_First, _Last) into ∗this after the iterator _Position. Insertion into a container after a specific position requires that the iterator be nonsingular (i.e., either dereferenceable or past-the-end), that it reference the sequence we are inserting into, and that the iterator range [_First, _Last) is a valid (possibly empty) range which does not reference the sequence we are inserting into. Note that this macro is only valid when the container is a _Safe_sequence and the _Position iterator is a _Safe_iterator.

__glibcxx_check_partitioned_lower

```
#define __glibcxx_check_partitioned_lower(
              _First,
              _Last,
              _Value)
```

Verify that the iterator range [_First, _Last) is partitioned w.r.t. the value _Value.

__glibcxx_check_partitioned_lower_pred

```
#define __glibcxx_check_partitioned_lower_pred(
              _First,
              _Last,
              _Value,
              _Pred)
```

Verify that the iterator range [_First, _Last) is partitioned w.r.t. the value _Value and predicate _Pred.

__glibcxx_check_partitioned_upper_pred

```
#define __glibcxx_check_partitioned_upper_pred(
              _First,
              _Last,
              _Value,
              _Pred)
```

Verify that the iterator range [_First, _Last) is partitioned w.r.t. the value _Value and predicate _Pred.

__glibcxx_check_sorted_pred

```
#define __glibcxx_check_sorted_pred(
              _First,
              _Last,
              _Pred)
```

Verify that the iterator range [_First, _Last) is sorted by the predicate _Pred.

_GLIBCXX_DEBUG_VERIFY_AT_F

```
#define _GLIBCXX_DEBUG_VERIFY_AT_F(
              _Cond,
              _ErrMsg,
              _File,
              _Line,
              _Func)
```

Macros used by the implementation to verify certain properties. These macros may only be used directly by the debug wrappers. Note that these are macros (instead of the more obviously *correct* choice of making them functions) because we need line and file information at the call site, to minimize the distance between the user error and where the error is reported.

6.257 map.h File Reference

Classes

- class std::__debug::map< _Key, _Tp, _Compare, _Allocator >

Namespaces

- namespace std
- namespace std::__debug

Functions

- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  std::__debug::map (_InputIterator, _InputIterator, _Allocator) -> map< __iter_key_t< _InputIterator >, __iter↩ _val_t< _InputIterator >, less< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Compare = less<__iter_key_t<_InputIterator>>, typename _Allocator = allocator<__iter↩ _to_alloc_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  std::__debug::map (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Compare, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::__debug::map (initializer_list< pair< _Key, _Tp > >, _Allocator) -> map< _Key, _Tp, less< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Compare = less<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  std::__debug::map (initializer_list< pair< _Key, _Tp > >, _Compare=_Compare(), _Allocator=_Allocator()) -> map< _Key, _Tp, _Compare, _Allocator >
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  __detail::__synth3way_t< pair< const _Key, _Tp > > **std::__debug::operator<=>** (const map< _Key, _Tp, _Compare, _Alloc > &__lhs, const map< _Key, _Tp, _Compare, _Alloc > &__rhs)

- template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>
  bool **std::__debug::operator==** (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>
  void **std::__debug::swap** (map< _Key, _Tp, _Compare, _Allocator > &__lhs, map< _Key, _Tp, _Compare, _Allocator > &__rhs) noexcept(/*conditional */)

6.257.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

6.258 multimap.h File Reference

Classes

- class std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >

Namespaces

- namespace std
- namespace std::__debug

Functions

- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  std::__debug::multimap (_InputIterator, _InputIterator, _Allocator) -> multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, less< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Compare = less<__iter_key_t<_InputIterator>>, typename _Allocator = allocator<__iter↩ _to_alloc_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  std::__debug::multimap (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Compare, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::__debug::multimap (initializer_list< pair< _Key, _Tp > >, _Allocator) -> multimap< _Key, _Tp, less< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Compare = less<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  std::__debug::multimap (initializer_list< pair< _Key, _Tp > >, _Compare=_Compare(), _Allocator=_↩ Allocator()) -> multimap< _Key, _Tp, _Compare, _Allocator >
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  __detail::__synth3way_t< pair< const _Key, _Tp > > **std::__debug::operator<=>** (const multimap< _Key, _Tp, _Compare, _Alloc > &__lhs, const multimap< _Key, _Tp, _Compare, _Alloc > &__rhs)
- template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>
  bool **std::__debug::operator==** (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>
  void **std::__debug::swap** (multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, multimap< _Key, _Tp, _↩ Compare, _Allocator > &__rhs) noexcept(/*conditional */)

6.258.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

6.259 multiset.h File Reference

Classes

- class std::__debug::multiset< _Key, _Compare, _Allocator >

Namespaces

- namespace std
- namespace std::__debug

Functions

- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  std::__debug::multiset (_InputIterator, _InputIterator, _Allocator) -> multiset< typename iterator_traits< _↩ InputIterator >::value_type, less< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Compare = less<typename iterator_traits<_InputIterator>::value_type>, typename _↩ Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  std::__debug::multiset (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> multiset< typename iterator_traits< _InputIterator >::value_type, _Compare, _Allocator >
- template<typename _Key, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::__debug::multiset (initializer_list< _Key >, _Allocator) -> multiset< _Key, less< _Key >, _Allocator >
- template<typename _Key, typename _Compare = less<_Key>, typename _Allocator = allocator<_Key>, typename = _RequireNot↩ Allocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  std::__debug::multiset (initializer_list< _Key >, _Compare=_Compare(), _Allocator=_Allocator()) -> multiset< _Key, _Compare, _Allocator >
- template<typename _Key, typename _Compare, typename _Alloc>
  __detail::__synth3way_t< _Key > **std::__debug::operator<=>** (const multiset< _Key, _Compare, _Alloc > &__lhs, const multiset< _Key, _Compare, _Alloc > &__rhs)
- template<typename _Key, typename _Compare, typename _Allocator>
  bool **std::__debug::operator==** (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key, typename _Compare, typename _Allocator>
  void **std::__debug::swap** (multiset< _Key, _Compare, _Allocator > &__x, multiset< _Key, _Compare, _Allocator > &__y) noexcept(/*conditional */)

6.259.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

6.260 safe_base.h File Reference

Classes

- class __gnu_debug::_Safe_iterator_base
- class __gnu_debug::_Safe_sequence_base

Namespaces

- namespace __gnu_debug

Functions

- bool __gnu_debug::__check_singular_aux (const _Safe_iterator_base *__x)

6.260.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

6.261 safe_container.h File Reference

Classes

- class __gnu_debug::_Safe_container< _SafeContainer, _Alloc, _SafeBase, _IsCxx11AllocatorAware >

Namespaces

- namespace __gnu_debug

6.261.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

6.262 safe_iterator.h File Reference

Classes

- struct __gnu_debug::_BeforeBeginHelper< _Sequence >
- class __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >
- struct __gnu_debug::_Sequence_traits< _Sequence >

Namespaces

- namespace __gnu_debug

Macros

- #define **_GLIBCXX20_CONSTEXPR_NON_LITERAL_SCOPE_BEGIN**
- #define **_GLIBCXX20_CONSTEXPR_NON_LITERAL_SCOPE_END**
- #define **_GLIBCXX_DEBUG_VERIFY_DIST_OPERANDS**(_Lhs, _Rhs)
- #define **_GLIBCXX_DEBUG_VERIFY_EQ_OPERANDS**(_Lhs, _Rhs)
- #define **_GLIBCXX_DEBUG_VERIFY_OPERANDS**(_Lhs, _Rhs, _BadMsgId, _DiffMsgId)
- #define **_GLIBCXX_DEBUG_VERIFY_REL_OPERANDS**(_Lhs, _Rhs)

Functions

- template<typename _Iterator, typename _Sequence>
  _Iterator **__gnu_debug::__base** (const _Safe_iterator< _Iterator, _Sequence, std::random_access_iterator_tag > &__it)
- template<typename _Iterator, typename _Sequence, typename _Category, typename _Size>
  bool **__gnu_debug::__can_advance** (const _Safe_iterator< _Iterator, _Sequence, _Category > &, _Size)
- template<typename _Iterator, typename _Sequence, typename _Category, typename _Diff>
  bool **__gnu_debug::__can_advance** (const _Safe_iterator< _Iterator, _Sequence, _Category > &, const std::pair< _Diff, _Distance_precision > &, int)
- template<typename _Iterator, typename _Sequence>
  _Iterator **__gnu_debug::__unsafe** (const _Safe_iterator< _Iterator, _Sequence > &__it)
- template<typename _Iterator, typename _Sequence, typename _Category>
  bool **__gnu_debug::__valid_range** (const _Safe_iterator< _Iterator, _Sequence, _Category > &, const _Safe_iterator< _Iterator, _Sequence, _Category > &)

- template<typename _Iterator, typename _Sequence, typename _Category>
  bool __gnu_debug::__valid_range (const _Safe_iterator< _Iterator, _Sequence, _Category > &, const _Safe_iterator< _Iterator, _Sequence, _Category > &, typename _Distance_traits< _Iterator >::__type &)

6.262.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

6.263 safe_iterator.tcc File Reference

Namespaces

- namespace __gnu_debug
- namespace std

Macros

- #define **_GLIBCXX_DEBUG_SAFE_ITERATOR_TCC**

Functions

- template<bool _IsMove, typename _II, typename _Ite, typename _Seq, typename _Cat>
  constexpr __gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > **std::__copy_move_a** (_II, _II, const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &)
- template<bool _IsMove, typename _IIte, typename _ISeq, typename _ICat, typename _OIte, typename _OSeq, typename _OCat>
  constexpr ::__gnu_debug::_Safe_iterator< _OIte, _OSeq, _OCat > **std::__copy_move_a** (const ::__gnu_debug::_Safe_iterator< _IIte, _ISeq, _ICat > &, const ::__gnu_debug::_Safe_iterator< _IIte, _ISeq, _ICat > &, const ::__gnu_debug::_Safe_iterator< _OIte, _OSeq, _OCat > &)
- template<bool _IsMove, typename _Ite, typename _Seq, typename _Cat, typename _OI>
  constexpr _OI **std::__copy_move_a** (const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &, const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &, _OI)
- template<bool _IsMove, typename _II, typename _Ite, typename _Seq, typename _Cat>
  constexpr __gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > **std::__copy_move_backward_a** (_II, _II, const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &)
- template<bool _IsMove, typename _IIte, typename _ISeq, typename _ICat, typename _OIte, typename _OSeq, typename _OCat>
  constexpr ::__gnu_debug::_Safe_iterator< _OIte, _OSeq, _OCat > **std::__copy_move_backward_a** (const ::__gnu_debug::_Safe_iterator< _IIte, _ISeq, _ICat > &, const ::__gnu_debug::_Safe_iterator< _IIte, _ISeq, _ICat > &, const ::__gnu_debug::_Safe_iterator< _OIte, _OSeq, _OCat > &)
- template<bool _IsMove, typename _Ite, typename _Seq, typename _Cat, typename _OI>
  constexpr _OI **std::__copy_move_backward_a** (const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &, const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &, _OI)
- template<typename _II1, typename _II2, typename _Seq2, typename _Cat2>
  constexpr bool **std::__equal_aux** (_II1, _II1, const ::__gnu_debug::_Safe_iterator< _II2, _Seq2, _Cat2 > &)
- template<typename _II1, typename _Seq1, typename _Cat1, typename _II2>
  constexpr bool **std::__equal_aux** (const ::__gnu_debug::_Safe_iterator< _II1, _Seq1, _Cat1 > &, const ::__gnu_debug::_Safe_iterator< _II1, _Seq1, _Cat1 > &, _II2)
- template<typename _II1, typename _Seq1, typename _Cat1, typename _II2, typename _Seq2, typename _Cat2>
  constexpr bool **std::__equal_aux** (const ::__gnu_debug::_Safe_iterator< _II1, _Seq1, _Cat1 > &, const ::__gnu_debug::_Safe_iterator< _II1, _Seq1, _Cat1 > &, const ::__gnu_debug::_Safe_iterator< _II2, _Seq2, _Cat2 > &)
- template<typename _Ite, typename _Seq, typename _Cat, typename _Tp>
  constexpr void **std::__fill_a** (const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &, const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &, const _Tp &)

- template<typename _Ite, typename _Seq, typename _Cat, typename _Size, typename _Tp>
  constexpr ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > **std::__fill_n_a** (const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &__first, _Size __n, const _Tp &__value, std::input_iterator_tag)
- template<typename _II1, typename _Ite2, typename _Seq2, typename _Cat2>
  constexpr bool **std::__lexicographical_compare_aux** (_II1 __first1, _II1 __last1, const ::__gnu_debug::_Safe_iterator< _Ite2, _Seq2, _Cat2 > &__first2, const ::__gnu_debug::_Safe_iterator< _Ite2, _Seq2, _Cat2 > &__last2)
- template<typename _Ite1, typename _Seq1, typename _Cat1, typename _II2>
  constexpr bool **std::__lexicographical_compare_aux** (const ::__gnu_debug::_Safe_iterator< _Ite1, _Seq1, _Cat1 > &__first1, const ::__gnu_debug::_Safe_iterator< _Ite1, _Seq1, _Cat1 > &__last1, _II2 __first2, _II2 __last2)
- template<typename _Ite1, typename _Seq1, typename _Cat1, typename _Ite2, typename _Seq2, typename _Cat2>
  constexpr bool **std::__lexicographical_compare_aux** (const ::__gnu_debug::_Safe_iterator< _Ite1, _Seq1, _Cat1 > &__first1, const ::__gnu_debug::_Safe_iterator< _Ite1, _Seq1, _Cat1 > &__last1, const ::__gnu_debug::_Safe_iterator< _Ite2, _Seq2, _Cat2 > &__first2, const ::__gnu_debug::_Safe_iterator< _Ite2, _Seq2, _Cat2 > &__last2)
- template<typename _Ite, typename _Seq>
  constexpr decltype(std::__niter_base(std::declval< _Ite >())) **std::__niter_base** (const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, std::random_access_iterator_tag > &__it) noexcept(std::is_nothrow_copy_constructible< _Ite >::value)

6.263.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

6.264 safe_local_iterator.h File Reference

Classes

- class __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >

Namespaces

- namespace __gnu_debug

Macros

- #define **_GLIBCXX_DEBUG_VERIFY_OPERANDS**(_Lhs, _Rhs)

Functions

- template<typename _Iterator, typename _UContainer>
  _Iterator **__gnu_debug::__unsafe** (const _Safe_local_iterator< _Iterator, _UContainer > &__it)
- template<typename _Iterator, typename _UContainer>
  bool **__gnu_debug::__valid_range** (const _Safe_local_iterator< _Iterator, _UContainer > &__first, const _Safe_local_iterator< _Iterator, _UContainer > &__last)
- template<typename _Iterator, typename _UContainer>
  bool __gnu_debug::__valid_range (const _Safe_local_iterator< _Iterator, _UContainer > &__first, const _Safe_local_iterator< _Iterator, _UContainer > &__last, typename _Distance_traits< _Iterator >::__type &__dist_info)

6.264.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

6.265 safe_local_iterator.tcc File Reference

Namespaces

- namespace __gnu_debug

Macros

- #define **_GLIBCXX_DEBUG_SAFE_LOCAL_ITERATOR_TCC**

6.265.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

6.266 safe_sequence.h File Reference

Classes

- class __gnu_debug::_After_nth_from< _Iterator >
- class __gnu_debug::_Equal_to< _Type >
- class __gnu_debug::_Not_equal_to< _Type >
- class __gnu_debug::_Safe_node_sequence< _Sequence >
- class __gnu_debug::_Safe_sequence< _Sequence >

Namespaces

- namespace __gnu_debug

6.266.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

6.267 safe_sequence.tcc File Reference

Namespaces

- namespace __gnu_debug

Macros

- #define **_GLIBCXX_DEBUG_SAFE_SEQUENCE_TCC**

6.267.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

6.268 safe_unordered_base.h File Reference

Classes

- class __gnu_debug::_Safe_local_iterator_base
- class __gnu_debug::_Safe_unordered_container_base

Namespaces

- namespace __gnu_debug

6.268.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

6.269 safe_unordered_container.h File Reference

Classes

- class __gnu_debug::_Safe_unordered_container< _Container >

Namespaces

- namespace __gnu_debug

6.269.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

6.270 safe_unordered_container.tcc File Reference

Namespaces

- namespace __gnu_debug

Macros

- #define **_GLIBCXX_DEBUG_SAFE_UNORDERED_CONTAINER_TCC**

6.270.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

6.271 set.h File Reference

Classes

- class std::__debug::set< _Key, _Compare, _Allocator >

Namespaces

- namespace std
- namespace std::__debug

Functions

- template<typename _Key, typename _Compare, typename _Alloc>
  __detail::__synth3way_t< _Key > **std::__debug::operator<=>** (const set< _Key, _Compare, _Alloc > &__lhs, const set< _Key, _Compare, _Alloc > &__rhs)
- template<typename _Key, typename _Compare, typename _Allocator>
  bool **std::__debug::operator==** (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  std::__debug::set (_InputIterator, _InputIterator, _Allocator) -> set< typename iterator_traits< _InputIterator >::value_type, less< typename iterator_traits< _InputIterator >::value_type >, _Allocator >

- template<typename _InputIterator, typename _Compare = less<typename iterator_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  std::__debug::set (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> set< typename iterator_traits< _InputIterator >::value_type, _Compare, _Allocator >
- template<typename _Key, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::__debug::set (initializer_list< _Key >, _Allocator) -> set< _Key, less< _Key >, _Allocator >
- template<typename _Key, typename _Compare = less<_Key>, typename _Allocator = allocator<_Key>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  std::__debug::set (initializer_list< _Key >, _Compare=_Compare(), _Allocator=_Allocator()) -> set< _Key, _Compare, _Allocator >
- template<typename _Key, typename _Compare, typename _Allocator>
  void **std::__debug::swap** (set< _Key, _Compare, _Allocator > &__x, set< _Key, _Compare, _Allocator > &__y) noexcept(/*conditional */)

6.271.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

6.272 decimal File Reference

Classes

- class std::decimal::decimal128
- class std::decimal::decimal32
- class std::decimal::decimal64

Namespaces

- namespace std
- namespace std::decimal

Macros

- #define **_DECLARE_DECIMAL128_COMPOUND_ASSIGNMENT**(_Op)
- #define **_DECLARE_DECIMAL32_COMPOUND_ASSIGNMENT**(_Op)
- #define **_DECLARE_DECIMAL64_COMPOUND_ASSIGNMENT**(_Op)
- #define **_DECLARE_DECIMAL_BINARY_OP_WITH_DEC**(_Op, _T1, _T2, _T3)
- #define **_DECLARE_DECIMAL_BINARY_OP_WITH_INT**(_Op, _Tp)
- #define **_DECLARE_DECIMAL_COMPARISON**(_Op, _Tp)
- #define **_GLIBCXX_DECIMAL**
- #define **_GLIBCXX_USE_DECIMAL_**

Functions

- double **std::decimal::decimal128_to_double** (decimal128 __d)
- float **std::decimal::decimal128_to_float** (decimal128 __d)
- long double **std::decimal::decimal128_to_long_double** (decimal128 __d)
- long long **std::decimal::decimal128_to_long_long** (decimal128 __d)
- double **std::decimal::decimal32_to_double** (decimal32 __d)
- float **std::decimal::decimal32_to_float** (decimal32 __d)
- long double **std::decimal::decimal32_to_long_double** (decimal32 __d)
- long long std::decimal::decimal32_to_long_long (decimal32 __d)
- double **std::decimal::decimal64_to_double** (decimal64 __d)

- float **std::decimal::decimal64_to_float** (decimal64 __d)
- long double **std::decimal::decimal64_to_long_double** (decimal64 __d)
- long long **std::decimal::decimal64_to_long_long** (decimal64 __d)
- double **std::decimal::decimal_to_double** (decimal128 __d)
- double **std::decimal::decimal_to_double** (decimal32 __d)
- double **std::decimal::decimal_to_double** (decimal64 __d)
- float **std::decimal::decimal_to_float** (decimal128 __d)
- float **std::decimal::decimal_to_float** (decimal32 __d)
- float **std::decimal::decimal_to_float** (decimal64 __d)
- long double **std::decimal::decimal_to_long_double** (decimal128 __d)
- long double **std::decimal::decimal_to_long_double** (decimal32 __d)
- long double **std::decimal::decimal_to_long_double** (decimal64 __d)
- long long **std::decimal::decimal_to_long_long** (decimal128 __d)
- long long **std::decimal::decimal_to_long_long** (decimal32 __d)
- long long **std::decimal::decimal_to_long_long** (decimal64 __d)
- static decimal128 **std::decimal::make_decimal128** (long long __coeff, int __exp)
- static decimal128 **std::decimal::make_decimal128** (unsigned long long __coeff, int __exp)
- static decimal32 **std::decimal::make_decimal32** (long long __coeff, int __exp)
- static decimal32 **std::decimal::make_decimal32** (unsigned long long __coeff, int __exp)
- static decimal64 **std::decimal::make_decimal64** (long long __coeff, int __exp)
- static decimal64 **std::decimal::make_decimal64** (unsigned long long __coeff, int __exp)
- bool **std::decimal::operator!=** (decimal128 __lhs, decimal128 __rhs)
- bool **std::decimal::operator!=** (decimal128 __lhs, decimal32 __rhs)
- bool **std::decimal::operator!=** (decimal128 __lhs, decimal64 __rhs)
- bool **std::decimal::operator!=** (decimal128 __lhs, int __rhs)
- bool **std::decimal::operator!=** (decimal128 __lhs, long __rhs)
- bool **std::decimal::operator!=** (decimal128 __lhs, long long __rhs)
- bool **std::decimal::operator!=** (decimal128 __lhs, unsigned int __rhs)
- bool **std::decimal::operator!=** (decimal128 __lhs, unsigned long __rhs)
- bool **std::decimal::operator!=** (decimal128 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator!=** (decimal32 __lhs, decimal128 __rhs)
- bool **std::decimal::operator!=** (decimal32 __lhs, decimal32 __rhs)
- bool **std::decimal::operator!=** (decimal32 __lhs, decimal64 __rhs)
- bool **std::decimal::operator!=** (decimal32 __lhs, int __rhs)
- bool **std::decimal::operator!=** (decimal32 __lhs, long __rhs)
- bool **std::decimal::operator!=** (decimal32 __lhs, long long __rhs)
- bool **std::decimal::operator!=** (decimal32 __lhs, unsigned int __rhs)
- bool **std::decimal::operator!=** (decimal32 __lhs, unsigned long __rhs)
- bool **std::decimal::operator!=** (decimal32 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator!=** (decimal64 __lhs, decimal128 __rhs)
- bool **std::decimal::operator!=** (decimal64 __lhs, decimal32 __rhs)
- bool **std::decimal::operator!=** (decimal64 __lhs, decimal64 __rhs)
- bool **std::decimal::operator!=** (decimal64 __lhs, int __rhs)
- bool **std::decimal::operator!=** (decimal64 __lhs, long __rhs)
- bool **std::decimal::operator!=** (decimal64 __lhs, long long __rhs)
- bool **std::decimal::operator!=** (decimal64 __lhs, unsigned int __rhs)
- bool **std::decimal::operator!=** (decimal64 __lhs, unsigned long __rhs)
- bool **std::decimal::operator!=** (decimal64 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator!=** (int __lhs, decimal128 __rhs)
- bool **std::decimal::operator!=** (int __lhs, decimal32 __rhs)
- bool **std::decimal::operator!=** (int __lhs, decimal64 __rhs)

- bool **std::decimal::operator!=** (long __lhs, decimal128 __rhs)
- bool **std::decimal::operator!=** (long __lhs, decimal32 __rhs)
- bool **std::decimal::operator!=** (long __lhs, decimal64 __rhs)
- bool **std::decimal::operator!=** (long long __lhs, decimal128 __rhs)
- bool **std::decimal::operator!=** (long long __lhs, decimal32 __rhs)
- bool **std::decimal::operator!=** (long long __lhs, decimal64 __rhs)
- bool **std::decimal::operator!=** (unsigned int __lhs, decimal128 __rhs)
- bool **std::decimal::operator!=** (unsigned int __lhs, decimal32 __rhs)
- bool **std::decimal::operator!=** (unsigned int __lhs, decimal64 __rhs)
- bool **std::decimal::operator!=** (unsigned long __lhs, decimal128 __rhs)
- bool **std::decimal::operator!=** (unsigned long __lhs, decimal32 __rhs)
- bool **std::decimal::operator!=** (unsigned long __lhs, decimal64 __rhs)
- bool **std::decimal::operator!=** (unsigned long long __lhs, decimal128 __rhs)
- bool **std::decimal::operator!=** (unsigned long long __lhs, decimal32 __rhs)
- bool **std::decimal::operator!=** (unsigned long long __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator*** (decimal128 __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator*** (decimal128 __lhs, decimal32 __rhs)
- decimal128 **std::decimal::operator*** (decimal128 __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator*** (decimal128 __lhs, int __rhs)
- decimal128 **std::decimal::operator*** (decimal128 __lhs, long __rhs)
- decimal128 **std::decimal::operator*** (decimal128 __lhs, long long __rhs)
- decimal128 **std::decimal::operator*** (decimal128 __lhs, unsigned int __rhs)
- decimal128 **std::decimal::operator*** (decimal128 __lhs, unsigned long __rhs)
- decimal128 **std::decimal::operator*** (decimal128 __lhs, unsigned long long __rhs)
- decimal128 **std::decimal::operator*** (decimal32 __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator*** (decimal32 __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator*** (decimal32 __lhs, decimal64 __rhs)
- decimal32 **std::decimal::operator*** (decimal32 __lhs, int __rhs)
- decimal32 **std::decimal::operator*** (decimal32 __lhs, long __rhs)
- decimal32 **std::decimal::operator*** (decimal32 __lhs, long long __rhs)
- decimal32 **std::decimal::operator*** (decimal32 __lhs, unsigned int __rhs)
- decimal32 **std::decimal::operator*** (decimal32 __lhs, unsigned long __rhs)
- decimal32 **std::decimal::operator*** (decimal32 __lhs, unsigned long long __rhs)
- decimal128 **std::decimal::operator*** (decimal64 __lhs, decimal128 __rhs)
- decimal64 **std::decimal::operator*** (decimal64 __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator*** (decimal64 __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator*** (decimal64 __lhs, int __rhs)
- decimal64 **std::decimal::operator*** (decimal64 __lhs, long __rhs)
- decimal64 **std::decimal::operator*** (decimal64 __lhs, long long __rhs)
- decimal64 **std::decimal::operator*** (decimal64 __lhs, unsigned int __rhs)
- decimal64 **std::decimal::operator*** (decimal64 __lhs, unsigned long __rhs)
- decimal64 **std::decimal::operator*** (decimal64 __lhs, unsigned long long __rhs)
- decimal128 **std::decimal::operator*** (int __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator*** (int __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator*** (int __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator*** (long __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator*** (long __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator*** (long __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator*** (long long __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator*** (long long __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator*** (long long __lhs, decimal64 __rhs)

- decimal128 **std::decimal::operator*** (unsigned int __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator*** (unsigned int __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator*** (unsigned int __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator*** (unsigned long __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator*** (unsigned long __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator*** (unsigned long __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator*** (unsigned long long __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator*** (unsigned long long __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator*** (unsigned long long __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator+** (decimal128 __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator+** (decimal128 __lhs, decimal32 __rhs)
- decimal128 **std::decimal::operator+** (decimal128 __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator+** (decimal128 __lhs, int __rhs)
- decimal128 **std::decimal::operator+** (decimal128 __lhs, long __rhs)
- decimal128 **std::decimal::operator+** (decimal128 __lhs, long long __rhs)
- decimal128 **std::decimal::operator+** (decimal128 __lhs, unsigned int __rhs)
- decimal128 **std::decimal::operator+** (decimal128 __lhs, unsigned long __rhs)
- decimal128 **std::decimal::operator+** (decimal128 __lhs, unsigned long long __rhs)
- decimal128 **std::decimal::operator+** (decimal128 __rhs)
- decimal128 **std::decimal::operator+** (decimal32 __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator+** (decimal32 __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator+** (decimal32 __lhs, decimal64 __rhs)
- decimal32 **std::decimal::operator+** (decimal32 __lhs, int __rhs)
- decimal32 **std::decimal::operator+** (decimal32 __lhs, long __rhs)
- decimal32 **std::decimal::operator+** (decimal32 __lhs, long long __rhs)
- decimal32 **std::decimal::operator+** (decimal32 __lhs, unsigned int __rhs)
- decimal32 **std::decimal::operator+** (decimal32 __lhs, unsigned long __rhs)
- decimal32 **std::decimal::operator+** (decimal32 __lhs, unsigned long long __rhs)
- decimal32 **std::decimal::operator+** (decimal32 __rhs)
- decimal128 **std::decimal::operator+** (decimal64 __lhs, decimal128 __rhs)
- decimal64 **std::decimal::operator+** (decimal64 __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator+** (decimal64 __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator+** (decimal64 __lhs, int __rhs)
- decimal64 **std::decimal::operator+** (decimal64 __lhs, long __rhs)
- decimal64 **std::decimal::operator+** (decimal64 __lhs, long long __rhs)
- decimal64 **std::decimal::operator+** (decimal64 __lhs, unsigned int __rhs)
- decimal64 **std::decimal::operator+** (decimal64 __lhs, unsigned long __rhs)
- decimal64 **std::decimal::operator+** (decimal64 __lhs, unsigned long long __rhs)
- decimal64 **std::decimal::operator+** (decimal64 __rhs)
- decimal128 **std::decimal::operator+** (int __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator+** (int __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator+** (int __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator+** (long __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator+** (long __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator+** (long __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator+** (long long __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator+** (long long __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator+** (long long __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator+** (unsigned int __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator+** (unsigned int __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator+** (unsigned int __lhs, decimal64 __rhs)

- decimal128 **std::decimal::operator+** (unsigned long __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator+** (unsigned long __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator+** (unsigned long __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator+** (unsigned long long __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator+** (unsigned long long __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator+** (unsigned long long __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator-** (decimal128 __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator-** (decimal128 __lhs, decimal32 __rhs)
- decimal128 **std::decimal::operator-** (decimal128 __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator-** (decimal128 __lhs, int __rhs)
- decimal128 **std::decimal::operator-** (decimal128 __lhs, long __rhs)
- decimal128 **std::decimal::operator-** (decimal128 __lhs, long long __rhs)
- decimal128 **std::decimal::operator-** (decimal128 __lhs, unsigned int __rhs)
- decimal128 **std::decimal::operator-** (decimal128 __lhs, unsigned long __rhs)
- decimal128 **std::decimal::operator-** (decimal128 __lhs, unsigned long long __rhs)
- decimal128 **std::decimal::operator-** (decimal128 __rhs)
- decimal128 **std::decimal::operator-** (decimal32 __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator-** (decimal32 __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator-** (decimal32 __lhs, decimal64 __rhs)
- decimal32 **std::decimal::operator-** (decimal32 __lhs, int __rhs)
- decimal32 **std::decimal::operator-** (decimal32 __lhs, long __rhs)
- decimal32 **std::decimal::operator-** (decimal32 __lhs, long long __rhs)
- decimal32 **std::decimal::operator-** (decimal32 __lhs, unsigned int __rhs)
- decimal32 **std::decimal::operator-** (decimal32 __lhs, unsigned long __rhs)
- decimal32 **std::decimal::operator-** (decimal32 __lhs, unsigned long long __rhs)
- decimal32 **std::decimal::operator-** (decimal32 __rhs)
- decimal128 **std::decimal::operator-** (decimal64 __lhs, decimal128 __rhs)
- decimal64 **std::decimal::operator-** (decimal64 __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator-** (decimal64 __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator-** (decimal64 __lhs, int __rhs)
- decimal64 **std::decimal::operator-** (decimal64 __lhs, long __rhs)
- decimal64 **std::decimal::operator-** (decimal64 __lhs, long long __rhs)
- decimal64 **std::decimal::operator-** (decimal64 __lhs, unsigned int __rhs)
- decimal64 **std::decimal::operator-** (decimal64 __lhs, unsigned long __rhs)
- decimal64 **std::decimal::operator-** (decimal64 __lhs, unsigned long long __rhs)
- decimal64 **std::decimal::operator-** (decimal64 __rhs)
- decimal128 **std::decimal::operator-** (int __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator-** (int __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator-** (int __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator-** (long __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator-** (long __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator-** (long __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator-** (long long __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator-** (long long __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator-** (long long __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator-** (unsigned int __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator-** (unsigned int __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator-** (unsigned int __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator-** (unsigned long __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator-** (unsigned long __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator-** (unsigned long __lhs, decimal64 __rhs)

- decimal128 **std::decimal::operator-** (unsigned long long __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator-** (unsigned long long __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator-** (unsigned long long __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator/** (decimal128 __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator/** (decimal128 __lhs, decimal32 __rhs)
- decimal128 **std::decimal::operator/** (decimal128 __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator/** (decimal128 __lhs, int __rhs)
- decimal128 **std::decimal::operator/** (decimal128 __lhs, long __rhs)
- decimal128 **std::decimal::operator/** (decimal128 __lhs, long long __rhs)
- decimal128 **std::decimal::operator/** (decimal128 __lhs, unsigned int __rhs)
- decimal128 **std::decimal::operator/** (decimal128 __lhs, unsigned long __rhs)
- decimal128 **std::decimal::operator/** (decimal128 __lhs, unsigned long long __rhs)
- decimal128 **std::decimal::operator/** (decimal32 __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator/** (decimal32 __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator/** (decimal32 __lhs, decimal64 __rhs)
- decimal32 **std::decimal::operator/** (decimal32 __lhs, int __rhs)
- decimal32 **std::decimal::operator/** (decimal32 __lhs, long __rhs)
- decimal32 **std::decimal::operator/** (decimal32 __lhs, long long __rhs)
- decimal32 **std::decimal::operator/** (decimal32 __lhs, unsigned int __rhs)
- decimal32 **std::decimal::operator/** (decimal32 __lhs, unsigned long __rhs)
- decimal32 **std::decimal::operator/** (decimal32 __lhs, unsigned long long __rhs)
- decimal128 **std::decimal::operator/** (decimal64 __lhs, decimal128 __rhs)
- decimal64 **std::decimal::operator/** (decimal64 __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator/** (decimal64 __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator/** (decimal64 __lhs, int __rhs)
- decimal64 **std::decimal::operator/** (decimal64 __lhs, long __rhs)
- decimal64 **std::decimal::operator/** (decimal64 __lhs, long long __rhs)
- decimal64 **std::decimal::operator/** (decimal64 __lhs, unsigned int __rhs)
- decimal64 **std::decimal::operator/** (decimal64 __lhs, unsigned long __rhs)
- decimal64 **std::decimal::operator/** (decimal64 __lhs, unsigned long long __rhs)
- decimal128 **std::decimal::operator/** (int __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator/** (int __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator/** (int __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator/** (long __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator/** (long __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator/** (long __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator/** (long long __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator/** (long long __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator/** (long long __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator/** (unsigned int __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator/** (unsigned int __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator/** (unsigned int __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator/** (unsigned long __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator/** (unsigned long __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator/** (unsigned long __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator/** (unsigned long long __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator/** (unsigned long long __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator/** (unsigned long long __lhs, decimal64 __rhs)
- bool **std::decimal::operator<** (decimal128 __lhs, decimal128 __rhs)
- bool **std::decimal::operator<** (decimal128 __lhs, decimal32 __rhs)
- bool **std::decimal::operator<** (decimal128 __lhs, decimal64 __rhs)

- bool **std::decimal::operator<** (decimal128 __lhs, int __rhs)
- bool **std::decimal::operator<** (decimal128 __lhs, long __rhs)
- bool **std::decimal::operator<** (decimal128 __lhs, long long __rhs)
- bool **std::decimal::operator<** (decimal128 __lhs, unsigned int __rhs)
- bool **std::decimal::operator<** (decimal128 __lhs, unsigned long __rhs)
- bool **std::decimal::operator<** (decimal128 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator<** (decimal32 __lhs, decimal128 __rhs)
- bool **std::decimal::operator<** (decimal32 __lhs, decimal32 __rhs)
- bool **std::decimal::operator<** (decimal32 __lhs, decimal64 __rhs)
- bool **std::decimal::operator<** (decimal32 __lhs, int __rhs)
- bool **std::decimal::operator<** (decimal32 __lhs, long __rhs)
- bool **std::decimal::operator<** (decimal32 __lhs, long long __rhs)
- bool **std::decimal::operator<** (decimal32 __lhs, unsigned int __rhs)
- bool **std::decimal::operator<** (decimal32 __lhs, unsigned long __rhs)
- bool **std::decimal::operator<** (decimal32 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator<** (decimal64 __lhs, decimal128 __rhs)
- bool **std::decimal::operator<** (decimal64 __lhs, decimal32 __rhs)
- bool **std::decimal::operator<** (decimal64 __lhs, decimal64 __rhs)
- bool **std::decimal::operator<** (decimal64 __lhs, int __rhs)
- bool **std::decimal::operator<** (decimal64 __lhs, long __rhs)
- bool **std::decimal::operator<** (decimal64 __lhs, long long __rhs)
- bool **std::decimal::operator<** (decimal64 __lhs, unsigned int __rhs)
- bool **std::decimal::operator<** (decimal64 __lhs, unsigned long __rhs)
- bool **std::decimal::operator<** (decimal64 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator<** (int __lhs, decimal128 __rhs)
- bool **std::decimal::operator<** (int __lhs, decimal32 __rhs)
- bool **std::decimal::operator<** (int __lhs, decimal64 __rhs)
- bool **std::decimal::operator<** (long __lhs, decimal128 __rhs)
- bool **std::decimal::operator<** (long __lhs, decimal32 __rhs)
- bool **std::decimal::operator<** (long __lhs, decimal64 __rhs)
- bool **std::decimal::operator<** (long long __lhs, decimal128 __rhs)
- bool **std::decimal::operator<** (long long __lhs, decimal32 __rhs)
- bool **std::decimal::operator<** (long long __lhs, decimal64 __rhs)
- bool **std::decimal::operator<** (unsigned int __lhs, decimal128 __rhs)
- bool **std::decimal::operator<** (unsigned int __lhs, decimal32 __rhs)
- bool **std::decimal::operator<** (unsigned int __lhs, decimal64 __rhs)
- bool **std::decimal::operator<** (unsigned long __lhs, decimal128 __rhs)
- bool **std::decimal::operator<** (unsigned long __lhs, decimal32 __rhs)
- bool **std::decimal::operator<** (unsigned long __lhs, decimal64 __rhs)
- bool **std::decimal::operator<** (unsigned long long __lhs, decimal128 __rhs)
- bool **std::decimal::operator<** (unsigned long long __lhs, decimal32 __rhs)
- bool **std::decimal::operator<** (unsigned long long __lhs, decimal64 __rhs)
- bool **std::decimal::operator==** (decimal128 __lhs, decimal128 __rhs)
- bool **std::decimal::operator==** (decimal128 __lhs, decimal32 __rhs)
- bool **std::decimal::operator==** (decimal128 __lhs, decimal64 __rhs)
- bool **std::decimal::operator==** (decimal128 __lhs, int __rhs)
- bool **std::decimal::operator==** (decimal128 __lhs, long __rhs)
- bool **std::decimal::operator==** (decimal128 __lhs, long long __rhs)
- bool **std::decimal::operator==** (decimal128 __lhs, unsigned int __rhs)
- bool **std::decimal::operator==** (decimal128 __lhs, unsigned long __rhs)
- bool **std::decimal::operator==** (decimal128 __lhs, unsigned long long __rhs)

- bool **std::decimal::operator==** (decimal32 __lhs, decimal128 __rhs)
- bool **std::decimal::operator==** (decimal32 __lhs, decimal32 __rhs)
- bool **std::decimal::operator==** (decimal32 __lhs, decimal64 __rhs)
- bool **std::decimal::operator==** (decimal32 __lhs, int __rhs)
- bool **std::decimal::operator==** (decimal32 __lhs, long __rhs)
- bool **std::decimal::operator==** (decimal32 __lhs, long long __rhs)
- bool **std::decimal::operator==** (decimal32 __lhs, unsigned int __rhs)
- bool **std::decimal::operator==** (decimal32 __lhs, unsigned long __rhs)
- bool **std::decimal::operator==** (decimal32 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator==** (decimal64 __lhs, decimal128 __rhs)
- bool **std::decimal::operator==** (decimal64 __lhs, decimal32 __rhs)
- bool **std::decimal::operator==** (decimal64 __lhs, decimal64 __rhs)
- bool **std::decimal::operator==** (decimal64 __lhs, int __rhs)
- bool **std::decimal::operator==** (decimal64 __lhs, long __rhs)
- bool **std::decimal::operator==** (decimal64 __lhs, long long __rhs)
- bool **std::decimal::operator==** (decimal64 __lhs, unsigned int __rhs)
- bool **std::decimal::operator==** (decimal64 __lhs, unsigned long __rhs)
- bool **std::decimal::operator==** (decimal64 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator==** (int __lhs, decimal128 __rhs)
- bool **std::decimal::operator==** (int __lhs, decimal32 __rhs)
- bool **std::decimal::operator==** (int __lhs, decimal64 __rhs)
- bool **std::decimal::operator==** (long __lhs, decimal128 __rhs)
- bool **std::decimal::operator==** (long __lhs, decimal32 __rhs)
- bool **std::decimal::operator==** (long __lhs, decimal64 __rhs)
- bool **std::decimal::operator==** (long long __lhs, decimal128 __rhs)
- bool **std::decimal::operator==** (long long __lhs, decimal32 __rhs)
- bool **std::decimal::operator==** (long long __lhs, decimal64 __rhs)
- bool **std::decimal::operator==** (unsigned int __lhs, decimal128 __rhs)
- bool **std::decimal::operator==** (unsigned int __lhs, decimal32 __rhs)
- bool **std::decimal::operator==** (unsigned int __lhs, decimal64 __rhs)
- bool **std::decimal::operator==** (unsigned long __lhs, decimal128 __rhs)
- bool **std::decimal::operator==** (unsigned long __lhs, decimal32 __rhs)
- bool **std::decimal::operator==** (unsigned long __lhs, decimal64 __rhs)
- bool **std::decimal::operator==** (unsigned long long __lhs, decimal128 __rhs)
- bool **std::decimal::operator==** (unsigned long long __lhs, decimal32 __rhs)
- bool **std::decimal::operator==** (unsigned long long __lhs, decimal64 __rhs)
- bool **std::decimal::operator>** (decimal128 __lhs, decimal128 __rhs)
- bool **std::decimal::operator>** (decimal128 __lhs, decimal32 __rhs)
- bool **std::decimal::operator>** (decimal128 __lhs, decimal64 __rhs)
- bool **std::decimal::operator>** (decimal128 __lhs, int __rhs)
- bool **std::decimal::operator>** (decimal128 __lhs, long __rhs)
- bool **std::decimal::operator>** (decimal128 __lhs, long long __rhs)
- bool **std::decimal::operator>** (decimal128 __lhs, unsigned int __rhs)
- bool **std::decimal::operator>** (decimal128 __lhs, unsigned long __rhs)
- bool **std::decimal::operator>** (decimal128 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator>** (decimal32 __lhs, decimal128 __rhs)
- bool **std::decimal::operator>** (decimal32 __lhs, decimal32 __rhs)
- bool **std::decimal::operator>** (decimal32 __lhs, decimal64 __rhs)
- bool **std::decimal::operator>** (decimal32 __lhs, int __rhs)
- bool **std::decimal::operator>** (decimal32 __lhs, long __rhs)
- bool **std::decimal::operator>** (decimal32 __lhs, long long __rhs)

- bool **std::decimal::operator**> (decimal32 __lhs, unsigned int __rhs)
- bool **std::decimal::operator**> (decimal32 __lhs, unsigned long __rhs)
- bool **std::decimal::operator**> (decimal32 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator**> (decimal64 __lhs, decimal128 __rhs)
- bool **std::decimal::operator**> (decimal64 __lhs, decimal32 __rhs)
- bool **std::decimal::operator**> (decimal64 __lhs, decimal64 __rhs)
- bool **std::decimal::operator**> (decimal64 __lhs, int __rhs)
- bool **std::decimal::operator**> (decimal64 __lhs, long __rhs)
- bool **std::decimal::operator**> (decimal64 __lhs, long long __rhs)
- bool **std::decimal::operator**> (decimal64 __lhs, unsigned int __rhs)
- bool **std::decimal::operator**> (decimal64 __lhs, unsigned long __rhs)
- bool **std::decimal::operator**> (decimal64 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator**> (int __lhs, decimal128 __rhs)
- bool **std::decimal::operator**> (int __lhs, decimal32 __rhs)
- bool **std::decimal::operator**> (int __lhs, decimal64 __rhs)
- bool **std::decimal::operator**> (long __lhs, decimal128 __rhs)
- bool **std::decimal::operator**> (long __lhs, decimal32 __rhs)
- bool **std::decimal::operator**> (long __lhs, decimal64 __rhs)
- bool **std::decimal::operator**> (long long __lhs, decimal128 __rhs)
- bool **std::decimal::operator**> (long long __lhs, decimal32 __rhs)
- bool **std::decimal::operator**> (long long __lhs, decimal64 __rhs)
- bool **std::decimal::operator**> (unsigned int __lhs, decimal128 __rhs)
- bool **std::decimal::operator**> (unsigned int __lhs, decimal32 __rhs)
- bool **std::decimal::operator**> (unsigned int __lhs, decimal64 __rhs)
- bool **std::decimal::operator**> (unsigned long __lhs, decimal128 __rhs)
- bool **std::decimal::operator**> (unsigned long __lhs, decimal32 __rhs)
- bool **std::decimal::operator**> (unsigned long __lhs, decimal64 __rhs)
- bool **std::decimal::operator**> (unsigned long long __lhs, decimal128 __rhs)
- bool **std::decimal::operator**> (unsigned long long __lhs, decimal32 __rhs)
- bool **std::decimal::operator**> (unsigned long long __lhs, decimal64 __rhs)
- bool **std::decimal::operator**>= (decimal128 __lhs, decimal128 __rhs)
- bool **std::decimal::operator**>= (decimal128 __lhs, decimal32 __rhs)
- bool **std::decimal::operator**>= (decimal128 __lhs, decimal64 __rhs)
- bool **std::decimal::operator**>= (decimal128 __lhs, int __rhs)
- bool **std::decimal::operator**>= (decimal128 __lhs, long __rhs)
- bool **std::decimal::operator**>= (decimal128 __lhs, long long __rhs)
- bool **std::decimal::operator**>= (decimal128 __lhs, unsigned int __rhs)
- bool **std::decimal::operator**>= (decimal128 __lhs, unsigned long __rhs)
- bool **std::decimal::operator**>= (decimal128 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator**>= (decimal32 __lhs, decimal128 __rhs)
- bool **std::decimal::operator**>= (decimal32 __lhs, decimal32 __rhs)
- bool **std::decimal::operator**>= (decimal32 __lhs, decimal64 __rhs)
- bool **std::decimal::operator**>= (decimal32 __lhs, int __rhs)
- bool **std::decimal::operator**>= (decimal32 __lhs, long __rhs)
- bool **std::decimal::operator**>= (decimal32 __lhs, long long __rhs)
- bool **std::decimal::operator**>= (decimal32 __lhs, unsigned int __rhs)
- bool **std::decimal::operator**>= (decimal32 __lhs, unsigned long __rhs)
- bool **std::decimal::operator**>= (decimal32 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator**>= (decimal64 __lhs, decimal128 __rhs)
- bool **std::decimal::operator**>= (decimal64 __lhs, decimal32 __rhs)
- bool **std::decimal::operator**>= (decimal64 __lhs, decimal64 __rhs)

- bool **std::decimal::operator>=** (decimal64 __lhs, int __rhs)
- bool **std::decimal::operator>=** (decimal64 __lhs, long __rhs)
- bool **std::decimal::operator>=** (decimal64 __lhs, long long __rhs)
- bool **std::decimal::operator>=** (decimal64 __lhs, unsigned int __rhs)
- bool **std::decimal::operator>=** (decimal64 __lhs, unsigned long __rhs)
- bool **std::decimal::operator>=** (decimal64 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator>=** (int __lhs, decimal128 __rhs)
- bool **std::decimal::operator>=** (int __lhs, decimal32 __rhs)
- bool **std::decimal::operator>=** (int __lhs, decimal64 __rhs)
- bool **std::decimal::operator>=** (long __lhs, decimal128 __rhs)
- bool **std::decimal::operator>=** (long __lhs, decimal32 __rhs)
- bool **std::decimal::operator>=** (long __lhs, decimal64 __rhs)
- bool **std::decimal::operator>=** (long long __lhs, decimal128 __rhs)
- bool **std::decimal::operator>=** (long long __lhs, decimal32 __rhs)
- bool **std::decimal::operator>=** (long long __lhs, decimal64 __rhs)
- bool **std::decimal::operator>=** (unsigned int __lhs, decimal128 __rhs)
- bool **std::decimal::operator>=** (unsigned int __lhs, decimal32 __rhs)
- bool **std::decimal::operator>=** (unsigned int __lhs, decimal64 __rhs)
- bool **std::decimal::operator>=** (unsigned long __lhs, decimal128 __rhs)
- bool **std::decimal::operator>=** (unsigned long __lhs, decimal32 __rhs)
- bool **std::decimal::operator>=** (unsigned long __lhs, decimal64 __rhs)
- bool **std::decimal::operator>=** (unsigned long long __lhs, decimal128 __rhs)
- bool **std::decimal::operator>=** (unsigned long long __lhs, decimal32 __rhs)
- bool **std::decimal::operator>=** (unsigned long long __lhs, decimal64 __rhs)

6.272.1 Detailed Description

This is a Standard C++ Library header.

6.273 debug/deque File Reference

Classes

- class std::__debug::deque< _Tp, _Allocator >

Namespaces

- namespace std
- namespace std::__debug

Macros

- #define **_GLIBCXX_DEBUG_DEQUE**

Functions

- template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  std::__debug::deque (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> deque< _ValT, _Allocator >
- template<typename _Tp, typename _Allocator = allocator<_Tp>, typename = _RequireAllocator<_Allocator>>
  std::__debug::deque (size_t, _Tp, _Allocator=_Allocator()) -> deque< _Tp, _Allocator >
- template<typename _Tp, typename _Alloc>
  constexpr __detail::__synth3way_t< _Tp > **std::__debug::operator<=>** (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)

- template<typename _Tp, typename _Alloc>
  bool **std::__debug::operator==** (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)
- template<typename _Tp, typename _Alloc>
  void **std::__debug::swap** (deque< _Tp, _Alloc > &__lhs, deque< _Tp, _Alloc > &__rhs) noexcept(/*conditional */)

6.273.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

6.274 deque File Reference

Namespaces

- namespace std

Macros

- #define **__glibcxx_want_algorithm_default_value_type**
- #define **__glibcxx_want_allocator_traits_is_always_equal**
- #define **__glibcxx_want_containers_ranges**
- #define **__glibcxx_want_erase_if**
- #define **__glibcxx_want_nonmember_container_access**
- #define **_GLIBCXX_DEQUE**

Typedefs

- template<typename _Tp>
  using **std::pmr::deque**

6.274.1 Detailed Description

This is a Standard C++ Library header.

6.275 experimental/deque File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **_GLIBCXX_EXPERIMENTAL_DEQUE**

Typedefs

- template<typename _Tp>
  using **std::experimental::fundamentals_v2::pmr::deque**

Functions

- template<typename _Tp, typename _Alloc, typename _Up>
  void **std::experimental::erase** (deque< _Tp, _Alloc > &__cont, const _Up &__value)
- template<typename _Tp, typename _Alloc, typename _Predicate>
  void **std::experimental::erase_if** (deque< _Tp, _Alloc > &__cont, _Predicate __pred)

6.275.1 Detailed Description

This is a TS C++ Library header.

6.276 expected File Reference

6.276.1 Detailed Description

This is a Standard C++ Library header.

6.277 lfts_config.h File Reference

6.277.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly.

6.278 ext/numeric_traits.h File Reference

Namespaces

- namespace __gnu_cxx

Macros

- #define **__glibcxx_digits10**(_Tp)
- #define **__glibcxx_floating**(_Tp, _Fval, _Dval, _LDval)
- #define **__glibcxx_max_digits10**(_Tp)
- #define **__glibcxx_max_exponent10**(_Tp)
- #define **_GLIBCXX_INT_N_TRAITS**(T, WIDTH)

Typedefs

- template<typename _Tp>
  using __gnu_cxx::__int_traits

Variables

- template<typename _Value>
  const int **__gnu_cxx::__numeric_traits_floating**< _Value >**::__digits10**
- template<typename _Value>
  const bool **__gnu_cxx::__numeric_traits_floating**< _Value >**::__is_signed**
- template<typename _Value>
  const int **__gnu_cxx::__numeric_traits_floating**< _Value >**::__max_digits10**
- template<typename _Value>
  const int **__gnu_cxx::__numeric_traits_floating**< _Value >**::__max_exponent10**
- template<typename _Value>
  const int **__gnu_cxx::__numeric_traits_integer**< _Value >**::__digits**
- template<typename _Value>
  const bool **__gnu_cxx::__numeric_traits_integer**< _Value >**::__is_signed**
- template<typename _Value>
  const _Value **__gnu_cxx::__numeric_traits_integer**< _Value >**::__max**
- template<typename _Value>
  const _Value **__gnu_cxx::__numeric_traits_integer**< _Value >**::__min**

6.278.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

6.279 propagate_const File Reference

Classes

- class std::experimental::fundamentals_v2::propagate_const< _Tp >

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **_GLIBCXX_EXPERIMENTAL_PROPAGATE_CONST**

Typedefs

- template<typename _Tp>
  using **std::experimental::__propagate_const_elem_type**

Functions

- template<typename _Tp>
  constexpr const _Tp & **std::experimental::get_underlying** (const propagate_const< _Tp > &__pt) noexcept
- template<typename _Tp>
  constexpr _Tp & **std::experimental::get_underlying** (propagate_const< _Tp > &__pt) noexcept
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator!=** (const _Tp &__t, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator!=** (const propagate_const< _Tp > &__pt, const _Up &__u)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator!=** (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)
- template<typename _Tp>
  constexpr bool **std::experimental::operator!=** (const propagate_const< _Tp > &__pt, nullptr_t)
- template<typename _Tp>
  constexpr bool **std::experimental::operator!=** (nullptr_t, const propagate_const< _Tp > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator**< (const _Tp &__t, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator**< (const propagate_const< _Tp > &__pt, const _Up &__u)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator**< (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator**<**=** (const _Tp &__t, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator**<**=** (const propagate_const< _Tp > &__pt, const _Up &__u)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator**<**=** (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)

- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator==** (const _Tp &__t, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator==** (const propagate_const< _Tp > &__pt, const _Up &__u)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator==** (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)
- template<typename _Tp>
  constexpr bool **std::experimental::operator==** (const propagate_const< _Tp > &__pt, nullptr_t)
- template<typename _Tp>
  constexpr bool **std::experimental::operator==** (nullptr_t, const propagate_const< _Tp > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator>** (const _Tp &__t, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator>** (const propagate_const< _Tp > &__pt, const _Up &__u)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator>** (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator>=** (const _Tp &__t, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator>=** (const propagate_const< _Tp > &__pt, const _Up &__u)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator>=** (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)
- template<typename _Tp>
  constexpr enable_if_t< __is_swappable< _Tp >::value, void > **std::experimental::swap** (propagate_const< _Tp > &__pt, propagate_const< _Tp > &__pt2) noexcept(__is_nothrow_swappable< _Tp >::value)

6.279.1 Detailed Description

This is a TS C++ Library header.

6.280 simd File Reference

Macros

- #define **__cpp_lib_experimental_parallel_simd**
- #define **_GLIBCXX_EXPERIMENTAL_SIMD**

6.280.1 Detailed Description

This is a TS C++ Library header.

6.281 aligned_buffer.h File Reference

Namespaces

- namespace __gnu_cxx

6.281.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

6.282 atomicity.h File Reference

Namespaces

- namespace __gnu_cxx

Macros

- #define **_GLIBCXX_READ_MEM_BARRIER**
- #define **_GLIBCXX_WRITE_MEM_BARRIER**

Functions

- void **__gnu_cxx::__atomic_add** (volatile _Atomic_word *__mem, int __val)
- void **__gnu_cxx::__atomic_add_dispatch** (_Atomic_word *__mem, int __val)
- void **__gnu_cxx::__atomic_add_single** (_Atomic_word *__mem, int __val)
- _Atomic_word **__gnu_cxx::__exchange_and_add** (volatile _Atomic_word *__mem, int __val)
- _Atomic_word **__gnu_cxx::__exchange_and_add_dispatch** (_Atomic_word *__mem, int __val)
- _Atomic_word **__gnu_cxx::__exchange_and_add_single** (_Atomic_word *__mem, int __val)
- bool **__gnu_cxx::__is_single_threaded** () noexcept

6.282.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

6.283 bitmap_allocator.h File Reference

Classes

- class __gnu_cxx::__detail::__mini_vector< _Tp >
- class __gnu_cxx::__detail::_Bitmap_counter< _Tp >
- class __gnu_cxx::__detail::_Ffit_finder< _Tp >
- class __gnu_cxx::bitmap_allocator< _Tp >
- class __gnu_cxx::free_list

Namespaces

- namespace __gnu_cxx
- namespace __gnu_cxx::__detail

Macros

- #define _BALLOC_ALIGN_BYTES

Enumerations

- enum { **bits_per_byte** , **bits_per_block** }

Functions

- void __gnu_cxx::__detail::__bit_allocate (std::size_t *__pbmap, std::size_t __pos) throw ()
- void __gnu_cxx::__detail::__bit_free (std::size_t *__pbmap, std::size_t __pos) throw ()
- template<typename _ForwardIterator, typename _Tp, typename _Compare>
  _ForwardIterator **__gnu_cxx::__detail::__lower_bound** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)

- template<typename _AddrPair>
  std::size_t __gnu_cxx::__detail::__num_bitmaps (_AddrPair __ap)
- template<typename _AddrPair>
  std::size_t __gnu_cxx::__detail::__num_blocks (_AddrPair __ap)
- std::size_t __gnu_cxx::_Bit_scan_forward (std::size_t __num)
- template<typename _Tp1, typename _Tp2>
  bool **__gnu_cxx::operator==** (const bitmap_allocator< _Tp1 > &, const bitmap_allocator< _Tp2 > &) throw ()

Variables

- template<typename _Tp>
  std::size_t **__gnu_cxx::bitmap_allocator< _Tp >::_S_block_size**
- template<typename _Tp>
  bitmap_allocator< _Tp >::_BPVector::size_type **__gnu_cxx::bitmap_allocator< _Tp >::_S_last_dealloc_index**
- template<typename _Tp>
  bitmap_allocator< _Tp >::_BPVector **__gnu_cxx::bitmap_allocator< _Tp >::_S_mem_blocks**
- template<typename _Tp>
  bitmap_allocator< _Tp >::__mutex_type **__gnu_cxx::bitmap_allocator< _Tp >::_S_mut**

6.283.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

6.283.2 Macro Definition Documentation

_BALLOC_ALIGN_BYTES

```
#define _BALLOC_ALIGN_BYTES
```

The constant in the expression below is the alignment required in bytes.

6.284 cast.h File Reference

Classes

- struct __gnu_cxx::_Caster< _ToType >

Namespaces

- namespace __gnu_cxx

Functions

- template<typename _ToType, typename _FromType>
  _ToType **__gnu_cxx::__const_pointer_cast** (_FromType *__arg)
- template<typename _ToType, typename _FromType>
  _ToType **__gnu_cxx::__const_pointer_cast** (const _FromType &__arg)
- template<typename _ToType, typename _FromType>
  _ToType **__gnu_cxx::__dynamic_pointer_cast** (_FromType *__arg)
- template<typename _ToType, typename _FromType>
  _ToType **__gnu_cxx::__dynamic_pointer_cast** (const _FromType &__arg)
- template<typename _ToType, typename _FromType>
  _ToType **__gnu_cxx::__reinterpret_pointer_cast** (_FromType *__arg)
- template<typename _ToType, typename _FromType>
  _ToType **__gnu_cxx::__reinterpret_pointer_cast** (const _FromType &__arg)

- template<typename _ToType, typename _FromType>
  _ToType __gnu_cxx::__static_pointer_cast (_FromType *__arg)
- template<typename _ToType, typename _FromType>
  _ToType __gnu_cxx::__static_pointer_cast (const _FromType &__arg)

6.284.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ext/pointer.h>.

6.285 codecvt_specializations.h File Reference

Classes

- class std::codecvt< _InternT, _ExternT, encoding_state >
- struct __gnu_cxx::encoding_char_traits< _CharT >
- class __gnu_cxx::encoding_state
- class std::encoding_state

Namespaces

- namespace __gnu_cxx
- namespace std

Functions

- template<typename _Tp>
  size_t **std::__iconv_adaptor** (size_t(*__func)(iconv_t, _Tp, size_t *, char **, size_t *), iconv_t __cd, char **__inbuf, size_t *__inbytes, char **__outbuf, size_t *__outbytes)

Variables

- template<typename _InternT, typename _ExternT>
  locale::id **std::codecvt< _InternT, _ExternT, encoding_state >::id**

6.285.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

6.286 concurrence.h File Reference

Classes

- class __gnu_cxx::__scoped_lock

Namespaces

- namespace __gnu_cxx

Enumerations

- enum **_Lock_policy** { **_S_single** , **_S_mutex** , **_S_atomic** }

Functions

- void **__gnu_cxx::__throw_concurrence_lock_error** ()
- void **__gnu_cxx::__throw_concurrence_unlock_error** ()

Variables

- const _Lock_policy **__gnu_cxx::__default_lock_policy**

6.286.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

6.287 debug_allocator.h File Reference

Classes

- class __gnu_cxx::debug_allocator< _Alloc >

Namespaces

- namespace __gnu_cxx

6.287.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

6.288 enc_filebuf.h File Reference

Classes

- class __gnu_cxx::enc_filebuf< _CharT >

Namespaces

- namespace __gnu_cxx

6.288.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

6.289 extptr_allocator.h File Reference

Classes

- class __gnu_cxx::_ExtPtr_allocator< _Tp >

Namespaces

- namespace __gnu_cxx

Functions

- template<typename _Tp>
  void **__gnu_cxx::swap** (_ExtPtr_allocator< _Tp > &__larg, _ExtPtr_allocator< _Tp > &__rarg)

6.289.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Author

Bob Walters

An example allocator which uses an alternative pointer type from bits/pointer.h. Supports test cases which confirm container support for alternative pointers.

6.290 malloc_allocator.h File Reference

Classes

- class __gnu_cxx::malloc_allocator< _Tp >

Namespaces

- namespace __gnu_cxx

6.290.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

6.291 mt_allocator.h File Reference

Classes

- struct __gnu_cxx::__common_pool_policy< _PoolTp, _Thread >
- class __gnu_cxx::__mt_alloc< _Tp, _Poolp >
- class __gnu_cxx::__mt_alloc_base< _Tp >
- struct __gnu_cxx::__per_type_pool_policy< _Tp, _PoolTp, _Thread >
- class __gnu_cxx::__pool< false >
- class __gnu_cxx::__pool< true >
- struct __gnu_cxx::__pool_base

Namespaces

- namespace __gnu_cxx

Macros

- #define **__thread_default**

Typedefs

- typedef void(* **__gnu_cxx::__destroy_handler**) (void *)

Functions

- template<typename _Tp, typename _Poolp>
  bool **__gnu_cxx::operator==** (const __mt_alloc< _Tp, _Poolp > &, const __mt_alloc< _Tp, _Poolp > &)

6.291.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

6.292 assoc_container.hpp File Reference

Classes

- class __gnu_pbds::basic_branch< Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc >
- class __gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc >
- class __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >
- class __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Al
- class __gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >
- class __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >
- class __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_BRANCH_BASE**
- #define **PB_DS_CC_HASH_BASE**
- #define **PB_DS_GP_HASH_BASE**
- #define **PB_DS_HASH_BASE**
- #define **PB_DS_LU_BASE**
- #define **PB_DS_TREE_BASE**
- #define **PB_DS_TREE_NODE_AND_IT_TRAITS**
- #define **PB_DS_TRIE_BASE**
- #define **PB_DS_TRIE_NODE_AND_IT_TRAITS**

6.292.1 Detailed Description

Contains associative containers.

6.293 bin_search_tree_.hpp File Reference

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_ASSERT_NODE_CONSISTENT**(_Node)
- #define **PB_DS_BIN_TREE_NAME**
- #define **PB_DS_BIN_TREE_TRAITS_BASE**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_STRUCT_ONLY_ASSERT_VALID**(X)

Variables

- template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>
  bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_allocator **__gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::s_node_allocator**

6.293.1 Detailed Description

Contains an implementation class for binary search tree.

6.294 bin_search_tree_/node_iterators.hpp File Reference

Classes

- class __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >
- class __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_TREE_CONST_NODE_ITERATOR_CLASS_C_DEC**
- #define **PB_DS_TREE_NODE_ITERATOR_CLASS_C_DEC**

6.294.1 Detailed Description

Contains an implementation class for bin_search_tree_.

6.295 ov_tree_map_/node_iterators.hpp File Reference

Classes

- class __gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >
- class __gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_OV_TREE_CONST_NODE_ITERATOR_C_DEC**
- #define **PB_DS_OV_TREE_NODE_ITERATOR_C_DEC**

6.295.1 Detailed Description

Contains an implementation class for ov_tree_.

6.296 point_iterators.hpp File Reference

Classes

- class __gnu_pbds::detail::bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Refe
- class __gnu_pbds::detail::bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference,

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_TREE_CONST_IT_C_DEC**
- #define **PB_DS_TREE_CONST_ODIR_IT_C_DEC**
- #define **PB_DS_TREE_IT_C_DEC**
- #define **PB_DS_TREE_ODIR_IT_C_DEC**

6.296.1 Detailed Description

Contains an implementation class for bin_search_tree_.

6.297 bin_search_tree_/r_erase_fn_imps.hpp File Reference

6.297.1 Detailed Description

Contains an implementation class for bin_search_tree_.

6.298 pat_trie_/r_erase_fn_imps.hpp File Reference

6.298.1 Detailed Description

Contains an implementation class for pat_trie.

6.299 bin_search_tree_/rotate_fn_imps.hpp File Reference

6.299.1 Detailed Description

Contains imps for rotating nodes.

6.300 pat_trie_/rotate_fn_imps.hpp File Reference

6.300.1 Detailed Description

Contains imps for rotating nodes.

6.301 bin_search_tree_/traits.hpp File Reference

Classes

- struct __gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >
- struct __gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >

Namespaces

- namespace __gnu_pbds

6.301.1 Detailed Description

Contains an implementation for bin_search_tree_.

6.302 branch_policy/traits.hpp File Reference

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_DEBUG_VERIFY**(_Cond)

6.302.1 Detailed Description

Contains an implementation class for tree-like classes.

6.303 ov_tree_map_/traits.hpp File Reference

Classes

- struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >
- struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >

Namespaces

- namespace __gnu_pbds

6.303.1 Detailed Description

Contains an implementation class for ov_tree_.

6.304 pat_trie_/traits.hpp File Reference

Classes

- struct __gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >
- struct __gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >

Namespaces

- namespace __gnu_pbds

6.304.1 Detailed Description

Contains an implementation class for pat_trie_.

6.305 rb_tree_map_/traits.hpp File Reference

Classes

- struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >
- struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >

Namespaces

- namespace __gnu_pbds

6.305.1 Detailed Description

Contains an implementation for rb_tree_.

6.306 splay_tree_/traits.hpp File Reference

Classes

- struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >
- struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >

Namespaces

- namespace __gnu_pbds

6.306.1 Detailed Description

Contains an implementation for splay_tree_.

6.307 binary_heap_.hpp File Reference

Classes

- class __gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_ASSERT_VALID**(X)
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_DEBUG_VERIFY**(_Cond)
- #define **PB_DS_ENTRY_CMP_DEC**
- #define **PB_DS_RESIZE_POLICY_DEC**

Variables

- template<typename Value_Type, typename Cmp_Fn, typename _Alloc>
  binary_heap< Value_Type, Cmp_Fn, _Alloc >::entry_allocator **__gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >::s_entry_allocator**
- template<typename Value_Type, typename Cmp_Fn, typename _Alloc>
  binary_heap< Value_Type, Cmp_Fn, _Alloc >::no_throw_copies_t **__gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >::s_no_throw_copies_ind**
- template<typename Value_Type, typename Cmp_Fn, typename _Alloc>
  binary_heap< Value_Type, Cmp_Fn, _Alloc >::value_allocator **__gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >::s_value_allocator**

6.307.1 Detailed Description

Contains an implementation class for a binary heap.

6.308 binary_heap_/const_iterator.hpp File Reference

Classes

- class __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_BIN_HEAP_CIT_BASE**

6.308.1 Detailed Description

Contains an iterator class returned by the table's const find and insert methods.

6.309 left_child_next_sibling_heap_/const_iterator.hpp File Reference

Classes

- class __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_BASIC_HEAP_CIT_BASE**
- #define **PB_DS_CLASS_C_DEC**

6.309.1 Detailed Description

Contains an iterator class returned by the table's const find and insert methods.

6.310 unordered_iterator/const_iterator.hpp File Reference

6.310.1 Detailed Description

Contains an iterator class used for const ranging over the elements of the table.
This file is intended to be included inside a class definition, with PB_DS_CLASS_C_DEC defined to the name of the enclosing class.

6.311 bin_search_tree_/constructors_destructor_fn_imps.hpp File Reference

6.311.1 Detailed Description

Contains an implementation class for bin_search_tree_.

6.312 binary_heap_/constructors_destructor_fn_imps.hpp File Reference

6.312.1 Detailed Description

Contains an implementation class for binary_heap_.

6.313 binomial_heap_/constructors_destructor_fn_imps.hpp File Reference

6.313.1 Detailed Description

Contains an implementation for binomial_heap_.

6.314 binomial_heap_base_/constructors_destructor_fn_imps.hpp File Reference

6.314.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

6.315 left_child_next_sibling_heap_/constructors_destructor_fn_imps.hpp File Reference

6.315.1 Detailed Description

Contains an implementation class for left_child_next_sibling_heap_.

6.316 ov_tree_map_/constructors_destructor_fn_imps.hpp File Reference

6.316.1 Detailed Description

Contains an implementation class for ov_tree_.

6.317 pairing_heap_/constructors_destructor_fn_imps.hpp File Reference

6.317.1 Detailed Description

Contains an implementation class for a pairing heap.

6.318 pat_trie_/constructors_destructor_fn_imps.hpp File Reference

6.318.1 Detailed Description

Contains an implementation class for pat_trie.

6.319 rb_tree_map_/constructors_destructor_fn_imps.hpp File Reference

6.319.1 Detailed Description

Contains an implementation for rb_tree_.

6.320 rc_binomial_heap_/constructors_destructor_fn_imps.hpp File Reference

6.320.1 Detailed Description

Contains an implementation for rc_binomial_heap_.

6.321 splay_tree_/constructors_destructor_fn_imps.hpp File Reference

6.321.1 Detailed Description

Contains an implementation class for splay_tree_.

6.322 thin_heap_/constructors_destructor_fn_imps.hpp File Reference

6.322.1 Detailed Description

Contains an implementation for thin_heap_.

6.323 bin_search_tree_/debug_fn_imps.hpp File Reference

6.323.1 Detailed Description

Contains an implementation class for bin_search_tree_.

6.324 binary_heap_/debug_fn_imps.hpp File Reference

6.324.1 Detailed Description

Contains an implementation class for a binary_heap.

6.325 binomial_heap_/debug_fn_imps.hpp File Reference

6.325.1 Detailed Description

Contains an implementation for binomial_heap_.

6.326 binomial_heap_base_/debug_fn_imps.hpp File Reference

6.326.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

6.327 cc_hash_table_map_/debug_fn_imps.hpp File Reference

6.327.1 Detailed Description

Contains implementations of cc_ht_map_'s debug-mode functions.

6.328 gp_hash_table_map_/debug_fn_imps.hpp File Reference

6.328.1 Detailed Description

Contains implementations of gp_ht_map_'s debug-mode functions.

6.329 left_child_next_sibling_heap_/debug_fn_imps.hpp File Reference

6.329.1 Detailed Description

Contains an implementation class for left_child_next_sibling_heap_.

6.330 list_update_map_/debug_fn_imps.hpp File Reference

6.330.1 Detailed Description

Contains implementations of cc_ht_map_'s debug-mode functions.

6.331 ov_tree_map_/debug_fn_imps.hpp File Reference

6.331.1 Detailed Description

Contains an implementation class for ov_tree_.

6.332 pairing_heap_/debug_fn_imps.hpp File Reference

6.332.1 Detailed Description

Contains an implementation class for a pairing heap.

6.333 pat_trie_/debug_fn_imps.hpp File Reference

6.333.1 Detailed Description

Contains an implementation class for pat_trie_.

6.334 rb_tree_map_/debug_fn_imps.hpp File Reference

6.334.1 Detailed Description

Contains an implementation for rb_tree_.

6.335 rc_binomial_heap_/debug_fn_imps.hpp File Reference

6.335.1 Detailed Description

Contains an implementation for rc_binomial_heap_.

6.336 splay_tree_/debug_fn_imps.hpp File Reference

6.336.1 Detailed Description

Contains an implementation class for splay_tree_.

6.337 thin_heap_/debug_fn_imps.hpp File Reference

6.337.1 Detailed Description

Contains an implementation for thin_heap_.

6.338 entry_cmp.hpp File Reference

Classes

- struct __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, false >
- struct __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, true >
- struct __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, false >::type

Namespaces

- namespace __gnu_pbds

6.338.1 Detailed Description

Contains an implementation class for a binary_heap.

6.339 entry_pred.hpp File Reference

Classes

- struct __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, false >
- struct __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, true >

Namespaces

- namespace __gnu_pbds

6.339.1 Detailed Description

Contains an implementation class for a binary_heap.

6.340 bin_search_tree_/erase_fn_imps.hpp File Reference

6.340.1 Detailed Description

Contains an implementation class for bin_search_tree_.

6.341 binary_heap_/erase_fn_imps.hpp File Reference

6.341.1 Detailed Description

Contains an implementation class for a binary_heap.

6.342 binomial_heap_base_/erase_fn_imps.hpp File Reference

6.342.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

6.343 cc_hash_table_map_/erase_fn_imps.hpp File Reference

6.343.1 Detailed Description

Contains implementations of cc_ht_map_'s erase related functions.

6.344 gp_hash_table_map_/erase_fn_imps.hpp File Reference

6.344.1 Detailed Description

Contains implementations of gp_ht_map_'s erase related functions.

6.345 left_child_next_sibling_heap_/erase_fn_imps.hpp File Reference

6.345.1 Detailed Description

Contains an implementation class for left_child_next_sibling_heap_.

6.346 list_update_map_/erase_fn_imps.hpp File Reference

6.346.1 Detailed Description

Contains implementations of lu_map_.

6.347 ov_tree_map_/erase_fn_imps.hpp File Reference

6.347.1 Detailed Description

Contains an implementation class for ov_tree_.

6.348 pairing_heap_/erase_fn_imps.hpp File Reference

6.348.1 Detailed Description

Contains an implementation class for a pairing heap.

6.349 pat_trie_/erase_fn_imps.hpp File Reference

6.349.1 Detailed Description

Contains an implementation class for pat_trie.

6.350 rb_tree_map_/erase_fn_imps.hpp File Reference

6.350.1 Detailed Description

Contains an implementation for rb_tree_.

6.351 rc_binomial_heap_/erase_fn_imps.hpp File Reference

6.351.1 Detailed Description

Contains an implementation for rc_binomial_heap_.

6.352 splay_tree_/erase_fn_imps.hpp File Reference

6.352.1 Detailed Description

Contains an implementation class for splay_tree_.

6.353 thin_heap_/erase_fn_imps.hpp File Reference

6.353.1 Detailed Description

Contains an implementation for thin_heap_.

6.354 bin_search_tree_/find_fn_imps.hpp File Reference

6.354.1 Detailed Description

Contains an implementation class for bin_search_tree_.

6.355 binary_heap_/find_fn_imps.hpp File Reference

6.355.1 Detailed Description

Contains an implementation class for a binary_heap.

6.356 binomial_heap_base_/find_fn_imps.hpp File Reference

6.356.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

6.357 cc_hash_table_map_/find_fn_imps.hpp File Reference

6.357.1 Detailed Description

Contains implementations of cc_ht_map_'s find related functions.

6.358 gp_hash_table_map_/find_fn_imps.hpp File Reference

6.358.1 Detailed Description

Contains implementations of gp_ht_map_'s find related functions.

6.359 list_update_map_/find_fn_imps.hpp File Reference

6.359.1 Detailed Description

Contains implementations of lu_map_.

6.360 pairing_heap_/find_fn_imps.hpp File Reference

6.360.1 Detailed Description

Contains an implementation class for a pairing heap.

6.361 pat_trie_/find_fn_imps.hpp File Reference

6.361.1 Detailed Description

Contains an implementation class for pat_trie.

6.362 rb_tree_map_/find_fn_imps.hpp File Reference

6.362.1 Detailed Description

Contains an implementation for rb_tree_.

6.363 splay_tree_/find_fn_imps.hpp File Reference

6.363.1 Detailed Description

Contains an implementation class for splay_tree_.

6.364 thin_heap_/find_fn_imps.hpp File Reference

6.364.1 Detailed Description

Contains an implementation for thin_heap_.

6.365 bin_search_tree_/info_fn_imps.hpp File Reference

6.365.1 Detailed Description

Contains an implementation class for bin_search_tree_.

6.366 binary_heap_/info_fn_imps.hpp File Reference

6.366.1 Detailed Description

Contains an implementation class for a binary_heap.

6.367 cc_hash_table_map_/info_fn_imps.hpp File Reference

6.367.1 Detailed Description

Contains implementations of cc_ht_map_'s entire container info related functions.

6.368 gp_hash_table_map_/info_fn_imps.hpp File Reference

6.368.1 Detailed Description

Contains implementations of gp_ht_map_'s entire container info related functions.

6.369 left_child_next_sibling_heap_/info_fn_imps.hpp File Reference

6.369.1 Detailed Description

Contains an implementation class for left_child_next_sibling_heap_.

6.370 list_update_map_/info_fn_imps.hpp File Reference

6.370.1 Detailed Description

Contains implementations of lu_map_.

6.371 ov_tree_map_/info_fn_imps.hpp File Reference

6.371.1 Detailed Description

Contains an implementation class for ov_tree_.

6.372 pat_trie_/info_fn_imps.hpp File Reference

6.372.1 Detailed Description

Contains an implementation class for pat_trie.

6.373 rb_tree_map_/info_fn_imps.hpp File Reference

6.373.1 Detailed Description

Contains an implementation for rb_tree_.

6.374 splay_tree_/info_fn_imps.hpp File Reference

6.374.1 Detailed Description

Contains an implementation.

6.375 bin_search_tree_/insert_fn_imps.hpp File Reference

6.375.1 Detailed Description

Contains an implementation class for bin_search_tree_.

6.376 binary_heap_/insert_fn_imps.hpp File Reference

6.376.1 Detailed Description

Contains an implementation class for a binary_heap.

6.377 binomial_heap_base_/insert_fn_imps.hpp File Reference

6.377.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

6.378 cc_hash_table_map_/insert_fn_imps.hpp File Reference

6.378.1 Detailed Description

Contains implementations of cc_ht_map_'s insert related functions.

6.379 gp_hash_table_map_/insert_fn_imps.hpp File Reference

6.379.1 Detailed Description

Contains implementations of gp_ht_map_'s insert related functions.

6.380 left_child_next_sibling_heap_/insert_fn_imps.hpp File Reference

6.380.1 Detailed Description

Contains an implementation class for left_child_next_sibling_heap_.

6.381 list_update_map_/insert_fn_imps.hpp File Reference

6.381.1 Detailed Description

Contains implementations of lu_map_.

6.382 ov_tree_map_/insert_fn_imps.hpp File Reference

6.382.1 Detailed Description

Contains an implementation class for ov_tree_.

6.383 pairing_heap_/insert_fn_imps.hpp File Reference

6.383.1 Detailed Description

Contains an implementation class for a pairing heap.

6.384 rb_tree_map_/insert_fn_imps.hpp File Reference

6.384.1 Detailed Description

Contains an implementation for rb_tree_.

6.385 rc_binomial_heap_/insert_fn_imps.hpp File Reference

6.385.1 Detailed Description

Contains an implementation for rc_binomial_heap_.

6.386 splay_tree_/insert_fn_imps.hpp File Reference

6.386.1 Detailed Description

Contains an implementation class for splay_tree_.

6.387 thin_heap_/insert_fn_imps.hpp File Reference

6.387.1 Detailed Description

Contains an implementation for thin_heap_.

6.388 bin_search_tree_/iterators_fn_imps.hpp File Reference

6.388.1 Detailed Description

Contains an implementation class for bin_search_tree_.

6.389 binary_heap_/iterators_fn_imps.hpp File Reference

6.389.1 Detailed Description

Contains an implementation class for a binary_heap.

6.390 cc_hash_table_map_/iterators_fn_imps.hpp File Reference

6.390.1 Detailed Description

Contains implementations of cc_ht_map_'s iterators related functions, e.g., begin().

6.391 left_child_next_sibling_heap_/iterators_fn_imps.hpp File Reference

6.391.1 Detailed Description

Contains an implementation class for left_child_next_sibling_heap_.

6.392 list_update_map_/iterators_fn_imps.hpp File Reference

6.392.1 Detailed Description

Contains implementations of lu_map_.

6.393 ov_tree_map_/iterators_fn_imps.hpp File Reference

6.393.1 Detailed Description

Contains an implementation class for ov_tree_.

6.394 pat_trie_/iterators_fn_imps.hpp File Reference

6.394.1 Detailed Description

Contains an implementation class for pat_trie.

6.395 binary_heap_/point_const_iterator.hpp File Reference

Classes

- class __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >

Namespaces

- namespace __gnu_pbds

6.395.1 Detailed Description

Contains an iterator class returned by the table's const find and insert methods.

6.396 left_child_next_sibling_heap_/point_const_iterator.hpp File Reference

Classes

- class __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**

6.396.1 Detailed Description

Contains an iterator class returned by the table's const find and insert methods.

6.397 unordered_iterator/point_const_iterator.hpp File Reference

6.397.1 Detailed Description

Contains an iterator class returned by the tables' const find and insert methods.

- This file is intended to be included inside a class definition, with PB_DS_CLASS_C_DEC defined to the name of the enclosing class.

6.398 bin_search_tree_/policy_access_fn_imps.hpp File Reference

6.398.1 Detailed Description

Contains an implementation class for bin_search_tree_.

6.399 binary_heap_/policy_access_fn_imps.hpp File Reference

6.399.1 Detailed Description

Contains an implementation class for a binary_heap.

6.400 cc_hash_table_map_/policy_access_fn_imps.hpp File Reference

6.400.1 Detailed Description

Contains implementations of cc_ht_map_'s policy access functions.

6.401 gp_hash_table_map_/policy_access_fn_imps.hpp File Reference

6.401.1 Detailed Description

Contains implementations of gp_ht_map_'s policy agpess functions.

6.402 left_child_next_sibling_heap_/policy_access_fn_imps.hpp File Reference

6.402.1 Detailed Description

Contains an implementation class for left_child_next_sibling_heap_.

6.403 ov_tree_map_/policy_access_fn_imps.hpp File Reference

6.403.1 Detailed Description

Contains an implementation class for ov_tree.

6.404 pat_trie_/policy_access_fn_imps.hpp File Reference

6.404.1 Detailed Description

Contains an implementation class for pat_trie.

6.405 resize_policy.hpp File Reference

Classes

- class __gnu_pbds::detail::resize_policy< _Tp >

Namespaces

- namespace __gnu_pbds

Variables

- template<typename _Tp>
  const _Tp **__gnu_pbds::detail::resize_policy< _Tp >::min_size**

6.405.1 Detailed Description

Contains an implementation class for a binary_heap.

6.406 bin_search_tree_/split_join_fn_imps.hpp File Reference

6.406.1 Detailed Description

Contains an implementation class for bin_search_tree_.

6.407 binary_heap_/split_join_fn_imps.hpp File Reference

6.407.1 Detailed Description

Contains an implementation class for a binary_heap.

6.408 binomial_heap_base_/split_join_fn_imps.hpp File Reference

6.408.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

6.409 ov_tree_map_/split_join_fn_imps.hpp File Reference

6.409.1 Detailed Description

Contains an implementation class for ov_tree_.

6.410 pairing_heap_/split_join_fn_imps.hpp File Reference

6.410.1 Detailed Description

Contains an implementation class for a pairing heap.

6.411 rb_tree_map_/split_join_fn_imps.hpp File Reference

6.411.1 Detailed Description

Contains an implementation for rb_tree_.

6.412 rc_binomial_heap_/split_join_fn_imps.hpp File Reference

6.412.1 Detailed Description

Contains an implementation for rc_binomial_heap_.

6.413 splay_tree_/split_join_fn_imps.hpp File Reference

6.413.1 Detailed Description

Contains an implementation class for splay_tree_.

6.414 thin_heap_/split_join_fn_imps.hpp File Reference

6.414.1 Detailed Description

Contains an implementation for thin_heap_.

6.415 binary_heap_/trace_fn_imps.hpp File Reference

6.415.1 Detailed Description

Contains an implementation class for a binary_heap.

6.416 cc_hash_table_map_/trace_fn_imps.hpp File Reference

6.416.1 Detailed Description

Contains implementations of cc_ht_map_'s trace-mode functions.

6.417 gp_hash_table_map_/trace_fn_imps.hpp File Reference

6.417.1 Detailed Description

Contains implementations of gp_ht_map_'s trace-mode functions.

6.418 left_child_next_sibling_heap_/trace_fn_imps.hpp File Reference

6.418.1 Detailed Description

Contains an implementation class for left_child_next_sibling_heap_.

6.419 list_update_map_/trace_fn_imps.hpp File Reference

6.419.1 Detailed Description

Contains implementations of lu_map_.

6.420 pat_trie_/trace_fn_imps.hpp File Reference

6.420.1 Detailed Description

Contains an implementation class for pat_trie_.

6.421 rc_binomial_heap_/trace_fn_imps.hpp File Reference

6.421.1 Detailed Description

Contains an implementation for rc_binomial_heap_.

6.422 thin_heap_/trace_fn_imps.hpp File Reference

6.422.1 Detailed Description

Contains an implementation class for left_child_next_sibling_heap_.

6.423 binomial_heap_.hpp File Reference

Classes

- class __gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**

6.423.1 Detailed Description

Contains an implementation class for a binomial heap.

6.424 binomial_heap_base_.hpp File Reference

Classes

- class __gnu_pbds::detail::binomial_heap_base< Value_Type, Cmp_Fn, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_ASSERT_BASE_NODE_CONSISTENT**(_Node, _Bool)
- #define **PB_DS_ASSERT_VALID_COND**(X, _StrictlyBinomial)
- #define **PB_DS_B_HEAP_BASE**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**

6.424.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

6.425 branch_policy.hpp File Reference

Classes

- struct __gnu_pbds::detail::branch_policy< Node_CItr, Node_Itr, _Alloc >
- struct __gnu_pbds::detail::branch_policy< Node_CItr, Node_CItr, _Alloc >

Namespaces

- namespace __gnu_pbds

6.425.1 Detailed Description

Contains a base class for branch policies.

6.426 null_node_metadata.hpp File Reference

Classes

- struct __gnu_pbds::detail::dumnode_const_iterator< Key, Data, _Alloc >

Namespaces

- namespace __gnu_pbds

6.426.1 Detailed Description

Contains an implementation class for tree-like classes.

6.427 cc_ht_map_.hpp File Reference

Classes

- class __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CC_HASH_NAME**
- #define **PB_DS_CC_HASH_TRAITS_BASE**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_GEN_POS**
- #define **PB_DS_HASH_EQ_FN_C_DEC**
- #define **PB_DS_RANGED_HASH_FN_C_DEC**

Variables

- template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb↩ _Hash_Fn, typename Resize_Policy>
  cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::const↩ _iterator **__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_↩ Hash_Fn, Resize_Policy >::s_const_end_it**
- template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb↩ _Hash_Fn, typename Resize_Policy>
  cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::iterator **__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::s_end_it**
- template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb↩ _Hash_Fn, typename Resize_Policy>
  cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::entry_↩ allocator **__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_↩ Hash_Fn, Resize_Policy >::s_entry_allocator**
- template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb↩ _Hash_Fn, typename Resize_Policy>
  cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::entry↩ _pointer_allocator **__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::s_entry_pointer_allocator**

6.427.1 Detailed Description

Contains an implementation class for cc_ht_map_.

6.428 cmp_fn_imps.hpp File Reference

6.428.1 Detailed Description

Contains implementations of cc_ht_map_'s entire container comparison related functions.

6.429 cond_key_dtor_entry_dealtor.hpp File Reference

Classes

- class __gnu_pbds::detail::cond_dealtor< Entry, _Alloc >

Namespaces

- namespace __gnu_pbds

6.429.1 Detailed Description

Contains a conditional key destructor, used for exception handling.

6.430 cc_hash_table_map_/constructor_destructor_fn_imps.hpp File Reference

6.430.1 Detailed Description

Contains implementations of cc_ht_map_'s constructors, destructor, and related functions.

6.431 gp_hash_table_map_/constructor_destructor_fn_imps.hpp File Reference

6.431.1 Detailed Description

Contains implementations of gp_ht_map_'s constructors, destructor, and related functions.

6.432 list_update_map_/constructor_destructor_fn_imps.hpp File Reference

6.433 cc_hash_table_map_/constructor_destructor_no_store_hash_fn_imps.hpp File Reference

6.433.1 Detailed Description

Contains implementations of cc_ht_map_'s constructors, destructor, and related functions.

6.434 gp_hash_table_map_/constructor_destructor_no_store_hash_fn_imps.hpp File Reference

6.434.1 Detailed Description

Contains implementations of gp_ht_map_'s constructors, destructor, and related functions.

6.435 cc_hash_table_map_/constructor_destructor_store_hash_fn_imps.hpp File Reference

6.435.1 Detailed Description

Contains implementations of cc_ht_map_'s constructors, destructor, and related functions.

6.436 gp_hash_table_map_/constructor_destructor_store_hash_fn_imps.hpp File Reference

6.436.1 Detailed Description

Contains implementations of gp_ht_map_'s constructors, destructor, and related functions.

6.437 cc_hash_table_map_/debug_no_store_hash_fn_imps.hpp File Reference

6.437.1 Detailed Description

Contains implementations of cc_ht_map_'s debug-mode functions.

6.438 gp_hash_table_map_/debug_no_store_hash_fn_imps.hpp File Reference

6.438.1 Detailed Description

Contains implementations of gp_ht_map_'s debug-mode functions.

6.439 cc_hash_table_map_/debug_store_hash_fn_imps.hpp File Reference

6.439.1 Detailed Description

Contains implementations of cc_ht_map_'s debug-mode functions.

6.440 gp_hash_table_map_/debug_store_hash_fn_imps.hpp File Reference

6.440.1 Detailed Description

Contains implementations of gp_ht_map_'s debug-mode functions.

6.441 entry_list_fn_imps.hpp File Reference

6.441.1 Detailed Description

Contains implementations of cc_ht_map_'s entry-list related functions.

6.442 cc_hash_table_map_/erase_no_store_hash_fn_imps.hpp File Reference

6.442.1 Detailed Description

Contains implementations of cc_ht_map_'s erase related functions, when the hash value is not stored.

6.443 gp_hash_table_map_/erase_no_store_hash_fn_imps.hpp File Reference

6.443.1 Detailed Description

Contains implementations of gp_ht_map_'s erase related functions, when the hash value is not stored.

6.444 cc_hash_table_map_/erase_store_hash_fn_imps.hpp File Reference

6.444.1 Detailed Description

Contains implementations of cc_ht_map_'s erase related functions, when the hash value is stored.

6.445 gp_hash_table_map_/erase_store_hash_fn_imps.hpp File Reference

6.445.1 Detailed Description

Contains implementations of gp_ht_map_'s erase related functions, when the hash value is stored.

6.446 cc_hash_table_map_/find_store_hash_fn_imps.hpp File Reference

6.446.1 Detailed Description

Contains implementations of cc_ht_map_'s find related functions, when the hash value is stored.

6.447 gp_hash_table_map_/find_store_hash_fn_imps.hpp File Reference

6.447.1 Detailed Description

Contains implementations of gp_ht_map_'s insert related functions, when the hash value is stored.

6.448 cc_hash_table_map_/insert_no_store_hash_fn_imps.hpp File Reference

6.448.1 Detailed Description

Contains implementations of cc_ht_map_'s insert related functions, when the hash value is not stored.

6.449 gp_hash_table_map_/insert_no_store_hash_fn_imps.hpp File Reference

6.449.1 Detailed Description

Contains implementations of gp_ht_map_'s insert related functions, when the hash value is not stored.

6.450 cc_hash_table_map_/insert_store_hash_fn_imps.hpp File Reference

6.450.1 Detailed Description

Contains implementations of cc_ht_map_'s insert related functions, when the hash value is stored.

6.451 gp_hash_table_map_/insert_store_hash_fn_imps.hpp File Reference

6.451.1 Detailed Description

Contains implementations of gp_ht_map_'s find related functions, when the hash value is stored.

6.452 cc_hash_table_map_/resize_fn_imps.hpp File Reference

6.452.1 Detailed Description

Contains implementations of cc_ht_map_'s resize related functions.

6.453 gp_hash_table_map_/resize_fn_imps.hpp File Reference

6.453.1 Detailed Description

Contains implementations of gp_ht_map_'s resize related functions.

6.454 cc_hash_table_map_/resize_no_store_hash_fn_imps.hpp File Reference

6.454.1 Detailed Description

Contains implementations of cc_ht_map_'s resize related functions, when the hash value is not stored.

6.455 gp_hash_table_map_/resize_no_store_hash_fn_imps.hpp File Reference

6.455.1 Detailed Description

Contains implementations of gp_ht_map_'s resize related functions, when the hash value is not stored.

6.456 cc_hash_table_map_/resize_store_hash_fn_imps.hpp File Reference

6.456.1 Detailed Description

Contains implementations of cc_ht_map_'s resize related functions, when the hash value is stored.

6.457 gp_hash_table_map_/resize_store_hash_fn_imps.hpp File Reference

6.457.1 Detailed Description

Contains implementations of gp_ht_map_'s resize related functions, when the hash value is stored.

6.458 size_fn_imps.hpp File Reference

6.458.1 Detailed Description

Contains implementations of cc_ht_map_'s entire container size related functions.

6.459 cond_dealtor.hpp File Reference

Classes

- class __gnu_pbds::detail::cond_dealtor< Entry, _Alloc >

Namespaces

- namespace __gnu_pbds

Variables

- template<typename Entry, class _Alloc>
  cond_dealtor< Entry, _Alloc >::entry_allocator **__gnu_pbds::detail::cond_dealtor< Entry, _Alloc >::s_alloc**

6.459.1 Detailed Description

Contains a conditional deallocator.

6.460 container_base_dispatch.hpp File Reference

Classes

- struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, cc_hash_tag, Policy_Tl >
- struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, gp_hash_tag, Policy_Tl >
- struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, list_update_tag, Policy_Tl >
- struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, ov_tree_tag, Policy_Tl >
- struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, pat_trie_tag, Policy_Tl >
- struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, rb_tree_tag, Policy_Tl >
- struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, splay_tree_tag, Policy_Tl >
- struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, cc_hash_tag, Policy_Tl >
- struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, gp_hash_tag, Policy_Tl >
- struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, list_update_tag, Policy_Tl >
- struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, ov_tree_tag, Policy_Tl >
- struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, pat_trie_tag, Policy_Tl >
- struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, rb_tree_tag, Policy_Tl >
- struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, splay_tree_tag, Policy_Tl >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_ASSERT_VALID**(X)
- #define **PB_DS_CHECK_KEY_DOES_NOT_EXIST**(_Key)
- #define **PB_DS_CHECK_KEY_EXISTS**(_Key)
- #define **PB_DS_DATA_FALSE_INDICATOR**
- #define **PB_DS_DATA_TRUE_INDICATOR**
- #define **PB_DS_DEBUG_VERIFY**(_Cond)
- #define **PB_DS_EP2VP**(X)
- #define **PB_DS_EP2VP**(X)
- #define **PB_DS_V2F**(X)
- #define **PB_DS_V2F**(X)
- #define **PB_DS_V2S**(X)
- #define **PB_DS_V2S**(X)

6.460.1 Detailed Description

Contains associative container dispatching.

6.461 debug_map_base.hpp File Reference

6.461.1 Detailed Description

Contains a debug-mode base for all maps.

6.462 eq_by_less.hpp File Reference

Classes

- struct __gnu_pbds::detail::eq_by_less< Key, Cmp_Fn >

Namespaces

- namespace __gnu_pbds

6.462.1 Detailed Description

Contains an equivalence function.

6.463 hash_eq_fn.hpp File Reference

Classes

- struct __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, false >
- struct __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, true >

Namespaces

- namespace __gnu_pbds

6.463.1 Detailed Description

Contains 2 equivalence functions, one employing a hash value, and one ignoring it.

6.464 find_no_store_hash_fn_imps.hpp File Reference

6.464.1 Detailed Description

Contains implementations of gp_ht_map_'s find related functions, when the hash value is not stored.

6.465 gp_ht_map_.hpp File Reference

Classes

- class __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_GEN_POS**
- #define **PB_DS_GP_HASH_NAME**
- #define **PB_DS_GP_HASH_TRAITS_BASE**
- #define **PB_DS_HASH_EQ_FN_C_DEC**
- #define **PB_DS_RANGED_PROBE_FN_C_DEC**

Variables

- template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb↩ _Probe_Fn, typename Probe_Fn, typename Resize_Policy>
  gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::const_iterator **__gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::s_const_end_it**
- template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb↩ _Probe_Fn, typename Probe_Fn, typename Resize_Policy>
  gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::iterator **__gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb↩ _Probe_Fn, Probe_Fn, Resize_Policy >::s_end_it**
- template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb↩ _Probe_Fn, typename Probe_Fn, typename Resize_Policy>
  gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::entry_allocator **__gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::s_entry_allocator**

6.465.1 Detailed Description

Contains an implementation class for general probing hash.

6.466 iterator_fn_imps.hpp File Reference

6.466.1 Detailed Description

Contains implementations of gp_ht_map_'s iterators related functions, e.g., begin().

6.467 direct_mask_range_hashing_imp.hpp File Reference

6.467.1 Detailed Description

Contains a range-hashing policy implementation

6.468 direct_mod_range_hashing_imp.hpp File Reference

6.468.1 Detailed Description

Contains a range-hashing policy implementation

6.469 linear_probe_fn_imp.hpp File Reference

6.469.1 Detailed Description

Contains a probe policy implementation

6.470 mask_based_range_hashing.hpp File Reference

Classes

- class __gnu_pbds::detail::mask_based_range_hashing< Size_Type >

Namespaces

- namespace __gnu_pbds

Variables

- template<typename Size_Type>
  const mask_based_range_hashing< Size_Type >::size_type **__gnu_pbds::detail::mask_based_range_hashing< Size_Type >::s_highest_bit_1**
- template<typename Size_Type>
  const mask_based_range_hashing< Size_Type >::size_type **__gnu_pbds::detail::mask_based_range_hashing< Size_Type >::s_num_bits_in_size_type**

6.470.1 Detailed Description

Contains a range hashing policy base.

6.471 mod_based_range_hashing.hpp File Reference

Classes

- class __gnu_pbds::detail::mod_based_range_hashing< Size_Type >

Namespaces

- namespace __gnu_pbds

6.471.1 Detailed Description

Contains a range hashing policy base.

6.472 probe_fn_base.hpp File Reference

Classes

- class __gnu_pbds::detail::probe_fn_base< _Alloc >

Namespaces

- namespace __gnu_pbds

6.472.1 Detailed Description

Contains a probe policy base.

6.473 quadratic_probe_fn_imp.hpp File Reference

6.473.1 Detailed Description

Contains a probe policy implementation

6.474 ranged_hash_fn.hpp File Reference

Classes

- class __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, false >
- class __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true >
- class __gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, false >
- class __gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, true >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CLASS_T_DEC**

6.474.1 Detailed Description

Contains a unified ranged hash functor, allowing the hash tables to deal with a single class for ranged hashing.

6.475 ranged_probe_fn.hpp File Reference

Classes

- class __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, false >
- class __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true >
- class __gnu_pbds::detail::ranged_probe_fn< Key, null_type, _Alloc, Comb_Probe_Fn, null_type, false >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CLASS_T_DEC**

6.475.1 Detailed Description

Contains a unified ranged probe functor, allowing the probe tables to deal with a single class for ranged probeing.

6.476 sample_probe_fn.hpp File Reference

Classes

- class __gnu_pbds::sample_probe_fn

Namespaces

- namespace __gnu_pbds

6.476.1 Detailed Description

Contains a sample probe policy.

6.477 sample_range_hashing.hpp File Reference

Classes

- class __gnu_pbds::sample_range_hashing

Namespaces

- namespace __gnu_pbds

6.477.1 Detailed Description

Contains a range hashing policy.

6.478 sample_ranged_hash_fn.hpp File Reference

Classes

- class __gnu_pbds::sample_ranged_hash_fn

Namespaces

- namespace __gnu_pbds

6.478.1 Detailed Description

Contains a ranged hash policy.

6.479 sample_ranged_probe_fn.hpp File Reference

Classes

- class __gnu_pbds::sample_ranged_probe_fn

Namespaces

- namespace __gnu_pbds

6.479.1 Detailed Description

Contains a ranged probe policy.

6.480 left_child_next_sibling_heap_.hpp File Reference

Classes

- class __gnu_pbds::detail::left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**

Variables

- template<typename Value_Type, typename Cmp_Fn, typename Node_Metadata, typename _Alloc>
  left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >::no_throw_copies_t **__gnu_pbds::detail::left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >::s_no_throw_copies_ind**
- template<typename Value_Type, typename Cmp_Fn, typename Node_Metadata, typename _Alloc>
  left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >::node_allocator **__gnu_pbds::detail::left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >::s_node_allocator**

6.480.1 Detailed Description

Contains an implementation class for a basic heap.

6.481 left_child_next_sibling_heap_/node.hpp File Reference

Classes

- struct __gnu_pbds::detail::left_child_next_sibling_heap_node_< _Value, _Metadata, _Alloc >

Namespaces

- namespace __gnu_pbds

6.481.1 Detailed Description

Contains an implementation struct for this type of heap's node.

6.482 rb_tree_map_/node.hpp File Reference

Classes

- struct __gnu_pbds::detail::rb_tree_node_< Value_Type, Metadata, _Alloc >

Namespaces

- namespace __gnu_pbds

6.482.1 Detailed Description

Contains an implementation for rb_tree_.

6.483 splay_tree_/node.hpp File Reference

Classes

- struct __gnu_pbds::detail::splay_tree_node_< Value_Type, Metadata, _Alloc >

Namespaces

- namespace __gnu_pbds

6.483.1 Detailed Description

Contains an implementation struct for splay_tree_'s node.

6.484 entry_metadata_base.hpp File Reference

Namespaces

- namespace __gnu_pbds

6.484.1 Detailed Description

Contains an implementation for a list update map.

6.485 lu_map_.hpp File Reference

Classes

- class __gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_GEN_POS**
- #define **PB_DS_LU_NAME**
- #define **PB_DS_LU_TRAITS_BASE**

Variables

- template<typename Key, typename Mapped, typename Eq_Fn, typename _Alloc, typename Update_Policy>
  lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >::entry_allocator **__gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >::s_entry_allocator**
- template<typename Key, typename Mapped, typename Eq_Fn, typename _Alloc, typename Update_Policy>
  Eq_Fn **__gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >::s_eq_fn**
- template<typename Key, typename Mapped, typename Eq_Fn, typename _Alloc, typename Update_Policy>
  type_to_type< typename lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >::update_metadata > __↩
  gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >::s_metadata_type_indicator
- template<typename Key, typename Mapped, typename Eq_Fn, typename _Alloc, typename Update_Policy>
  null_type **__gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >::s_null_type**
- template<typename Key, typename Mapped, typename Eq_Fn, typename _Alloc, typename Update_Policy>
  Update_Policy **__gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >::s_update_↩
  policy**

6.485.1 Detailed Description

Contains a list update map.

6.486 lu_counter_metadata.hpp File Reference

Classes

- class __gnu_pbds::detail::lu_counter_metadata< Size_Type >
- class __gnu_pbds::detail::lu_counter_policy_base< Size_Type >

Namespaces

- namespace __gnu_pbds

6.486.1 Detailed Description

Contains implementation of a lu counter policy's metadata.

6.487 sample_update_policy.hpp File Reference

Classes

- struct __gnu_pbds::sample_update_policy

Namespaces

- namespace __gnu_pbds

6.487.1 Detailed Description

Contains a sample policy for list update containers.

6.488 ov_tree_map_.hpp File Reference

Classes

- class __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::cond_dtor< Size_Type >
- class __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CONST_NODE_ITERATOR_NAME**
- #define **PB_DS_OV_TREE_NAME**
- #define **PB_DS_OV_TREE_TRAITS_BASE**

Variables

- template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc> ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::metadata_allocator **__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::s_metadata_alloc**
- template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc> ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::value_allocator **__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::s_value_alloc**

6.488.1 Detailed Description

Contains an implementation class for ov_tree.

6.489 pairing_heap_.hpp File Reference

Classes

- class __gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_ASSERT_NODE_CONSISTENT**(_Node, _Bool)
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_P_HEAP_BASE**

6.489.1 Detailed Description

Contains an implementation class for a pairing heap.

6.490 insert_join_fn_imps.hpp File Reference

6.490.1 Detailed Description

Contains an implementation class for pat_trie.

6.491 pat_trie_.hpp File Reference

Classes

- class __gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_ASSERT_NODE_VALID**(X)
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_PAT_TRIE_NAME**
- #define **PB_DS_PAT_TRIE_TRAITS_BASE**
- #define **PB_DS_RECURSIVE_COUNT_LEAFS**(X)

Variables

- template<typename Key, typename Mapped, typename Node_And_It_Traits, typename _Alloc>
  pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::head_allocator **__gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::s_head_allocator**
- template<typename Key, typename Mapped, typename Node_And_It_Traits, typename _Alloc>
  pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::inode_allocator **__gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::s_inode_allocator**
- template<typename Key, typename Mapped, typename Node_And_It_Traits, typename _Alloc>
  pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::leaf_allocator **__gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::s_leaf_allocator**

6.491.1 Detailed Description

Contains an implementation class for a patricia tree.

6.492 pat_trie_base.hpp File Reference

Classes

- class __gnu_pbds::detail::pat_trie_base::_CIter< Node, Leaf, Head, Inode, Is_Forward_Iterator >
- struct __gnu_pbds::detail::pat_trie_base::_Head< _ATraits, Metadata >
- struct __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >
- class __gnu_pbds::detail::pat_trie_base::_Iter< Node, Leaf, Head, Inode, Is_Forward_Iterator >
- struct __gnu_pbds::detail::pat_trie_base::_Leaf< _ATraits, Metadata >
- struct __gnu_pbds::detail::pat_trie_base::_Metadata< Metadata, _Alloc >
- struct __gnu_pbds::detail::pat_trie_base::_Metadata< null_type, _Alloc >
- struct __gnu_pbds::detail::pat_trie_base::_Node_base< _ATraits, Metadata >
- class __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >
- class __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >
- struct __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >::const_iterator
- struct __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >::iterator
- struct __gnu_pbds::detail::pat_trie_base

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CONST_IT_C_DEC**
- #define **PB_DS_CONST_ODIR_IT_C_DEC**
- #define **PB_DS_IT_C_DEC**
- #define **PB_DS_ODIR_IT_C_DEC**
- #define **PB_DS_PAT_TRIE_NODE_CONST_ITERATOR_C_DEC**
- #define **PB_DS_PAT_TRIE_NODE_ITERATOR_C_DEC**

6.492.1 Detailed Description

Contains the base class for a patricia tree.

6.493 split_fn_imps.hpp File Reference

6.493.1 Detailed Description

Contains an implementation class for pat_trie.

6.494 synth_access_traits.hpp File Reference

Classes

- struct __gnu_pbds::detail::synth_access_traits< Type_Traits, Set, _ATraits >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_SYNTH_E_ACCESS_TRAITS_C_DEC**
- #define **PB_DS_SYNTH_E_ACCESS_TRAITS_T_DEC**

Variables

- template<typename Type_Traits, bool Set, typename _ATraits>
  integral_constant< int, Set > **__gnu_pbds::detail::synth_access_traits< Type_Traits, Set, _ATraits >::s_set_ind**

6.494.1 Detailed Description

Contains an implementation class for a patricia tree.

6.495 update_fn_imps.hpp File Reference

6.495.1 Detailed Description

Contains an implementation class for pat_trie_.

6.496 priority_queue_base_dispatch.hpp File Reference

Classes

- struct __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binary_heap_tag, null_type >
- struct __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binomial_heap_tag, null_type >
- struct __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, pairing_heap_tag, null_type >
- struct __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type >
- struct __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, thin_heap_tag, null_type >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_ASSERT_VALID**(X)
- #define **PB_DS_DEBUG_VERIFY**(_Cond)

6.496.1 Detailed Description

Contains an pqiative container dispatching base.

6.497 rb_tree_.hpp File Reference

Classes

- class __gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_RB_TREE_BASE**
- #define **PB_DS_RB_TREE_BASE_NAME**
- #define **PB_DS_RB_TREE_NAME**
- #define **PB_DS_STRUCT_ONLY_ASSERT_VALID**(X)

6.497.1 Detailed Description

Contains an implementation for Red Black trees.

6.498 rc.hpp File Reference

Classes

- class __gnu_pbds::detail::rc< _Node, _Alloc >

Namespaces

- namespace __gnu_pbds

6.498.1 Detailed Description

Contains a redundant (binary counter).

6.499 rc_binomial_heap_.hpp File Reference

Classes

- class __gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_RC_C_DEC**

6.499.1 Detailed Description

Contains an implementation for redundant-counter binomial heap.

6.500 cc_hash_max_collision_check_resize_trigger_imp.hpp File Reference

6.500.1 Detailed Description

Contains a resize trigger implementation.

6.501 hash_exponential_size_policy_imp.hpp File Reference

6.501.1 Detailed Description

Contains a resize size policy implementation.

6.502 hash_load_check_resize_trigger_imp.hpp File Reference

6.502.1 Detailed Description

Contains a resize trigger implementation.

6.503 hash_load_check_resize_trigger_size_base.hpp File Reference

Classes

- class __gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, true >

Namespaces

- namespace __gnu_pbds

6.503.1 Detailed Description

Contains an base holding size for some resize policies.

6.504 hash_prime_size_policy_imp.hpp File Reference

6.504.1 Detailed Description

Contains a resize size policy implementation.

6.505 hash_standard_resize_policy_imp.hpp File Reference

6.505.1 Detailed Description

Contains a resize policy implementation.

6.506 sample_resize_policy.hpp File Reference

Classes

- class __gnu_pbds::sample_resize_policy

Namespaces

- namespace __gnu_pbds

6.506.1 Detailed Description

Contains a sample resize policy for hash tables.

6.507 sample_resize_trigger.hpp File Reference

Classes

- class __gnu_pbds::sample_resize_trigger

Namespaces

- namespace __gnu_pbds

6.507.1 Detailed Description

Contains a sample resize trigger policy class.

6.508 sample_size_policy.hpp File Reference

Classes

- class __gnu_pbds::sample_size_policy

Namespaces

- namespace __gnu_pbds

6.508.1 Detailed Description

Contains a sample size resize-policy.

6.509 splay_fn_imps.hpp File Reference

6.509.1 Detailed Description

Contains an implementation class for splay_tree_.

6.510 splay_tree_.hpp File Reference

Classes

- class __gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_ASSERT_BASE_NODE_CONSISTENT**(_Node)
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_S_TREE_BASE**
- #define **PB_DS_S_TREE_BASE_NAME**
- #define **PB_DS_S_TREE_NAME**

6.510.1 Detailed Description

Contains an implementation class for splay trees.

6.511 standard_policies.hpp File Reference

Classes

- struct __gnu_pbds::detail::default_comb_hash_fn
- struct __gnu_pbds::detail::default_eq_fn< Key >
- struct __gnu_pbds::detail::default_hash_fn< Key >
- struct __gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn >
- struct __gnu_pbds::detail::default_resize_policy< Comb_Hash_Fn >
- struct __gnu_pbds::detail::default_trie_access_traits< std::basic_string< Char, Char_Traits, std::allocator< char > > >
- struct __gnu_pbds::detail::default_update_policy

Namespaces

- namespace __gnu_pbds

Macros

- #define **__dtrie_alloc**
- #define **__dtrie_string**

Enumerations

- enum { **default_store_hash** }

6.511.1 Detailed Description

Contains standard policies for containers.

6.511.2 Enumeration Type Documentation

anonymous enum

`anonymous enum`
Enumeration for default behavior of stored hash data.

6.512 thin_heap_.hpp File Reference

Classes

- class __gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_ASSERT_AUX_NULL**(X)
- #define **PB_DS_ASSERT_NODE_CONSISTENT**(_Node, _Bool)
- #define **PB_DS_BASE_T_P**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**

Enumerations

- enum { **num_distinct_rank_bounds** }

Variables

- static const std::size_t **__gnu_pbds::detail::g_a_rank_bounds** [num_distinct_rank_bounds]

6.512.1 Detailed Description

Contains an implementation class for a thin heap.

6.513 tree_policy/node_metadata_selector.hpp File Reference

Classes

- struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, false >
- struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, true >
- struct __gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >

Namespaces

- namespace __gnu_pbds

6.513.1 Detailed Description

Contains an implementation class for trees.

6.514 trie_policy/node_metadata_selector.hpp File Reference

Classes

- struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, false >
- struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, true >
- struct __gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >

Namespaces

- namespace __gnu_pbds

6.514.1 Detailed Description

Contains an implementation class for tries.

6.515 tree_policy/order_statistics_imp.hpp File Reference

6.515.1 Detailed Description

Contains forward declarations for order_statistics_key

6.516 trie_policy/order_statistics_imp.hpp File Reference

6.516.1 Detailed Description

Contains forward declarations for order_statistics_key

6.517 sample_tree_node_update.hpp File Reference

Classes

- class __gnu_pbds::sample_tree_node_update< Const_Node_Iter, Node_Iter, Cmp_Fn, _Alloc >

Namespaces

- namespace __gnu_pbds

6.517.1 Detailed Description

Contains a samle node update functor.

6.518 tree_trace_base.hpp File Reference

6.518.1 Detailed Description

Contains tree-related policies.

6.519 prefix_search_node_update_imp.hpp File Reference

6.519.1 Detailed Description

Contains an implementation of prefix_search_node_update.

6.520 sample_trie_access_traits.hpp File Reference

Classes

- struct __gnu_pbds::sample_trie_access_traits

Namespaces

- namespace __gnu_pbds

6.520.1 Detailed Description

Contains a sample probe policy.

6.521 sample_trie_node_update.hpp File Reference

Classes

- class __gnu_pbds::sample_trie_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >

Namespaces

- namespace __gnu_pbds

6.521.1 Detailed Description

Contains a samle node update functor.

6.522 trie_policy_base.hpp File Reference

Classes

- class __gnu_pbds::detail::trie_policy_base< Node_CItr, Node_Itr, _ATraits, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**

6.522.1 Detailed Description

Contains an implementation of trie_policy_base.

6.523 trie_string_access_traits_imp.hpp File Reference

6.523.1 Detailed Description

Contains a policy for extracting character positions from a string for a vector-based PATRICIA tree

6.524 type_utils.hpp File Reference

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_STATIC_ASSERT**(UNIQUE, E)

Typedefs

- typedef std::tr1::integral_constant< int, 0 > **__gnu_pbds::detail::false_type**
- typedef std::tr1::integral_constant< int, 1 > **__gnu_pbds::detail::true_type**

6.524.1 Detailed Description

Contains utilities for handling types. All of these classes are based on Modern C++ by Andrei Alxandrescu.

6.525 types_traits.hpp File Reference

Classes

- struct __gnu_pbds::detail::maybe_null_type< Key, Mapped, _Alloc, Store_Hash >
- struct __gnu_pbds::detail::maybe_null_type< Key, null_type, _Alloc, Store_Hash >
- struct __gnu_pbds::detail::no_throw_copies< Key, Mapped >
- struct __gnu_pbds::detail::no_throw_copies< Key, null_type >
- struct __gnu_pbds::detail::rebind_traits< _Alloc, T >
- struct __gnu_pbds::detail::select_value_type< Key, Mapped >
- struct __gnu_pbds::detail::select_value_type< Key, null_type >
- struct __gnu_pbds::detail::stored_data< _Tv, _Th, Store_Hash >
- struct __gnu_pbds::detail::stored_data< _Tv, _Th, false >
- struct __gnu_pbds::detail::stored_hash< _Th >
- struct __gnu_pbds::detail::stored_value< _Tv >
- struct __gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash >

Namespaces

- namespace __gnu_pbds

Variables

- template<typename Key, typename _Alloc, bool Store_Hash>
  null_type **__gnu_pbds::detail::maybe_null_type< Key, null_type, _Alloc, Store_Hash >::s_null_type**

6.525.1 Detailed Description

Contains a traits class of types used by containers.

6.526 iterator.hpp File Reference

6.526.1 Detailed Description

Contains an iterator_ class used for ranging over the elements of the table.
This file is intended to be included inside a class definition, with PB_DS_CLASS_C_DEC defined to the name of the enclosing class.

6.527 point_iterator.hpp File Reference

6.527.1 Detailed Description

Contains an iterator class returned by the tables' find and insert methods.
This file is intended to be included inside a class definition, with PB_DS_CLASS_C_DEC defined to the name of the enclosing class.

6.528 exception.hpp File Reference

Classes

- struct __gnu_pbds::container_error
- struct __gnu_pbds::insert_error
- struct __gnu_pbds::join_error
- struct __gnu_pbds::resize_error

Namespaces

- namespace __gnu_pbds

Functions

- void **__gnu_pbds::__throw_container_error** ()
- void **__gnu_pbds::__throw_insert_error** ()
- void **__gnu_pbds::__throw_join_error** ()
- void **__gnu_pbds::__throw_resize_error** ()

6.528.1 Detailed Description

Contains exception classes.

6.529 hash_policy.hpp File Reference

Classes

- class __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >
- class __gnu_pbds::direct_mask_range_hashing< Size_Type >
- class __gnu_pbds::direct_mod_range_hashing< Size_Type >
- class __gnu_pbds::hash_exponential_size_policy< Size_Type >
- class __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >
- class __gnu_pbds::hash_prime_size_policy
- class __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >
- class __gnu_pbds::linear_probe_fn< Size_Type >
- class __gnu_pbds::quadratic_probe_fn< Size_Type >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_ASSERT_VALID**(X)
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_SIZE_BASE_C_DEC**

Enumerations

- enum { **num_distinct_sizes_16_bit** , **num_distinct_sizes_32_bit** , **num_distinct_sizes_64_bit** , **num_distinct_sizes** }

Variables

- static const std::size_t **__gnu_pbds::detail::g_a_sizes** [num_distinct_sizes]

6.529.1 Detailed Description

Contains hash-related policies.

6.530 list_update_policy.hpp File Reference

Classes

- class __gnu_pbds::lu_counter_policy< Max_Count, _Alloc >
- class __gnu_pbds::lu_move_to_front_policy< _Alloc >

Namespaces

- namespace __gnu_pbds

6.530.1 Detailed Description

Contains policies for list update containers.

6.531 priority_queue.hpp File Reference

Classes

- class __gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >

Namespaces

- namespace __gnu_pbds

6.531.1 Detailed Description

Contains priority_queues.

6.532 tag_and_trait.hpp File Reference

Classes

- struct __gnu_pbds::associative_tag
- struct __gnu_pbds::basic_branch_tag
- struct __gnu_pbds::basic_hash_tag
- struct __gnu_pbds::basic_invalidation_guarantee
- struct __gnu_pbds::binary_heap_tag
- struct __gnu_pbds::binomial_heap_tag
- struct __gnu_pbds::cc_hash_tag
- struct __gnu_pbds::container_tag
- struct __gnu_pbds::container_traits< Cntnr >
- struct __gnu_pbds::container_traits_base< binary_heap_tag >
- struct __gnu_pbds::container_traits_base< binomial_heap_tag >
- struct __gnu_pbds::container_traits_base< cc_hash_tag >
- struct __gnu_pbds::container_traits_base< gp_hash_tag >
- struct __gnu_pbds::container_traits_base< list_update_tag >
- struct __gnu_pbds::container_traits_base< ov_tree_tag >
- struct __gnu_pbds::container_traits_base< pairing_heap_tag >
- struct __gnu_pbds::container_traits_base< pat_trie_tag >
- struct __gnu_pbds::container_traits_base< rb_tree_tag >
- struct __gnu_pbds::container_traits_base< rc_binomial_heap_tag >
- struct __gnu_pbds::container_traits_base< splay_tree_tag >
- struct __gnu_pbds::container_traits_base< thin_heap_tag >
- struct __gnu_pbds::gp_hash_tag
- struct __gnu_pbds::list_update_tag
- struct __gnu_pbds::null_node_update< _Tp1, _Tp2, _Tp3, _Tp4 >
- struct __gnu_pbds::null_type
- struct __gnu_pbds::ov_tree_tag
- struct __gnu_pbds::pairing_heap_tag
- struct __gnu_pbds::pat_trie_tag
- struct __gnu_pbds::point_invalidation_guarantee
- struct __gnu_pbds::priority_queue_tag
- struct __gnu_pbds::range_invalidation_guarantee
- struct __gnu_pbds::rb_tree_tag
- struct __gnu_pbds::rc_binomial_heap_tag
- struct __gnu_pbds::sequence_tag
- struct __gnu_pbds::splay_tree_tag
- struct __gnu_pbds::string_tag
- struct __gnu_pbds::thin_heap_tag
- struct __gnu_pbds::tree_tag
- struct __gnu_pbds::trie_tag
- struct __gnu_pbds::trivial_iterator_tag

Namespaces

- namespace __gnu_pbds

Typedefs

- typedef void __gnu_pbds::trivial_iterator_difference_type

6.532.1 Detailed Description

Contains tags and traits, e.g., ones describing underlying data structures.

6.533 tree_policy.hpp File Reference

Classes

- class __gnu_pbds::tree_order_statistics_node_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_BRANCH_POLICY_BASE**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**

6.533.1 Detailed Description

Contains tree-related policies.

6.534 trie_policy.hpp File Reference

Classes

- class __gnu_pbds::trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >
- class __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >
- struct __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_TRIE_POLICY_BASE**

Variables

- template<typename String, typename String::value_type Min_E_Val, typename String::value_type Max_E_Val, bool Reverse, typename _Alloc>
  detail::integral_constant< int, Reverse > **__gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::s_rev_ind**

6.534.1 Detailed Description

Contains trie-related policies.

6.535 pod_char_traits.h File Reference

Classes

- struct std::char_traits< __gnu_cxx::character< _Value, _Int, _St > >
- struct __gnu_cxx::character< _Value, _Int, _St >

Namespaces

- namespace __gnu_cxx
- namespace std

Functions

- template<typename _Value, typename _Int, typename _St>
  bool **__gnu_cxx::operator<** (const character< _Value, _Int, _St > &lhs, const character< _Value, _Int, _St > &rhs)
- template<typename _Value, typename _Int, typename _St>
  bool **__gnu_cxx::operator==** (const character< _Value, _Int, _St > &lhs, const character< _Value, _Int, _St > &rhs)

6.535.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

6.536 pointer.h File Reference

Classes

- struct __gnu_cxx::_Invalid_type
- class __gnu_cxx::_Pointer_adapter< _Storage_policy >
- class __gnu_cxx::_Relative_pointer_impl< _Tp >
- class __gnu_cxx::_Relative_pointer_impl< const _Tp >
- class __gnu_cxx::_Std_pointer_impl< _Tp >
- struct __gnu_cxx::_Unqualified_type< _Tp >

Namespaces

- namespace __gnu_cxx
- namespace std

Macros

- #define **_CXX_POINTER_ARITH_OPERATOR_SET**(INT_TYPE)
- #define **_GCC_CXX_POINTER_COMPARISON_OPERATION_SET**(OPERATOR)

Functions

- template<typename _Tp1, typename _Tp2>
  bool **__gnu_cxx::operator!=** (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _Tp>
  bool **__gnu_cxx::operator!=** (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)
- template<typename _Tp>
  bool **__gnu_cxx::operator!=** (const _Pointer_adapter< _Tp > &__lhs, int __rhs)
- template<typename _Tp1, typename _Tp2>
  bool **__gnu_cxx::operator!=** (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)
- template<typename _Tp1, typename _Tp2>
  bool **__gnu_cxx::operator!=** (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &_↩
  _rhs)
- template<typename _Tp>
  bool **__gnu_cxx::operator!=** (int __lhs, const _Pointer_adapter< _Tp > &__rhs)
- template<typename _Tp1, typename _Tp2>
  bool **__gnu_cxx::operator<** (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _Tp1, typename _Tp2>
  bool **__gnu_cxx::operator<** (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)
- template<typename _Tp1, typename _Tp2>
  bool **__gnu_cxx::operator<** (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &_↩
  _rhs)
- template<typename _CharT, typename _Traits, typename _StoreT>
  std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &↩
  __os, const _Pointer_adapter< _StoreT > &__p)
- template<typename _Tp1, typename _Tp2>
  bool **__gnu_cxx::operator<=** (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _Tp>
  bool **__gnu_cxx::operator<=** (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__↩
  rhs)
- template<typename _Tp1, typename _Tp2>
  bool **__gnu_cxx::operator<=** (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)
- template<typename _Tp1, typename _Tp2>
  bool **__gnu_cxx::operator<=** (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &↩
  __rhs)
- template<typename _Tp1, typename _Tp2>
  bool **__gnu_cxx::operator==** (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _Tp>
  bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)
- template<typename _Tp>
  bool **__gnu_cxx::operator==** (const _Pointer_adapter< _Tp > &__lhs, int __rhs)
- template<typename _Tp1, typename _Tp2>
  bool **__gnu_cxx::operator==** (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)
- template<typename _Tp1, typename _Tp2>
  bool **__gnu_cxx::operator==** (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &↩
  __rhs)
- template<typename _Tp>
  bool **__gnu_cxx::operator==** (int __lhs, const _Pointer_adapter< _Tp > &__rhs)
- template<typename _Tp1, typename _Tp2>
  bool **__gnu_cxx::operator>** (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _Tp>
  bool **__gnu_cxx::operator>** (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)

- template<typename _Tp1, typename _Tp2>
  bool **__gnu_cxx::operator>** (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)
- template<typename _Tp1, typename _Tp2>
  bool **__gnu_cxx::operator>** (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &_↩
  _rhs)
- template<typename _Tp1, typename _Tp2>
  bool **__gnu_cxx::operator>=** (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _Tp>
  bool **__gnu_cxx::operator>=** (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__↩
  rhs)
- template<typename _Tp1, typename _Tp2>
  bool **__gnu_cxx::operator>=** (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)
- template<typename _Tp1, typename _Tp2>
  bool **__gnu_cxx::operator>=** (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &↩
  __rhs)

6.536.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Author

Bob Walters

Provides reusable _Pointer_adapter for assisting in the development of custom pointer types that can be used with the standard containers via the allocator::pointer and allocator::const_pointer typedefs.

6.537 pool_allocator.h File Reference

Classes

- class __gnu_cxx::__pool_alloc< _Tp >
- class __gnu_cxx::__pool_alloc_base

Namespaces

- namespace __gnu_cxx

Functions

- template<typename _Tp>
  bool **__gnu_cxx::operator==** (const __pool_alloc< _Tp > &, const __pool_alloc< _Tp > &)

Variables

- template<typename _Tp>
  _Atomic_word **__gnu_cxx::__pool_alloc< _Tp >::_S_force_new**

6.537.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

6.538 rb_tree File Reference

Classes

- struct __gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc >

Namespaces

- namespace __gnu_cxx

Macros

- #define **_RB_TREE**

6.538.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

6.539 rc_string_base.h File Reference

Classes

- class __gnu_cxx::__rc_string_base< _CharT, _Traits, _Alloc >

Namespaces

- namespace __gnu_cxx

Variables

- template<typename _CharT, typename _Traits, typename _Alloc>
  __rc_string_base< _CharT, _Traits, _Alloc >::_Rep_empty **__gnu_cxx::__rc_string_base< _CharT, _Traits, _Alloc >::_S_empty_rep**

6.539.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ext/vstring.h>.

6.540 rope File Reference

Classes

- class __gnu_cxx::rope< _CharT, _Alloc >

Namespaces

- namespace __gnu_cxx
- namespace __gnu_cxx::__detail
- namespace std
- namespace std::tr1

Macros

- #define **__GC_CONST**
- #define **__ROPE_DEFINE_ALLOC**(_Tp, __name)
- #define **__ROPE_DEFINE_ALLOC**(_Tp, __name)
- #define **__ROPE_DEFINE_ALLOCS**(__a)
- #define **__STATIC_IF_SGI_ALLOC**
- #define **__STL_FREE_STRING**(__s, __l, __a)
- #define **__STL_ROPE_FROM_UNOWNED_CHAR_PTR**(__s, __size, __a)
- #define **_ROPE**

Typedefs

- typedef rope< char > **__gnu_cxx::crope**
- typedef rope< wchar_t > **__gnu_cxx::wrope**

Enumerations

- enum { **_S_max_rope_depth** }
- enum **_Tag** { **_S_leaf** , **_S_concat** , **_S_substringfn** , **_S_function** }

Functions

- crope::reference **__gnu_cxx::__mutable_reference_at** (crope &__c, std::size_t __i)
- template<typename _ForwardIterator, typename _Allocator>
  void **__gnu_cxx::_Destroy_const** (_ForwardIterator __first, _ForwardIterator __last, _Allocator __alloc)
- template<typename _ForwardIterator, typename _Tp>
  void **__gnu_cxx::_Destroy_const** (_ForwardIterator __first, _ForwardIterator __last, std::allocator< _Tp >)
- template<class _CharT>
  void **__gnu_cxx::_S_cond_store_eos** (_CharT &)
- void **__gnu_cxx::_S_cond_store_eos** (char &__c)
- void **__gnu_cxx::_S_cond_store_eos** (wchar_t &__c)
- template<class _CharT>
  _CharT **__gnu_cxx::_S_eos** (_CharT *)
- template<class _CharT>
  bool **__gnu_cxx::_S_is_basic_char_type** (_CharT *)
- bool **__gnu_cxx::_S_is_basic_char_type** (char *)
- bool **__gnu_cxx::_S_is_basic_char_type** (wchar_t *)
- template<class _CharT>
  bool **__gnu_cxx::_S_is_one_byte_char_type** (_CharT *)
- bool **__gnu_cxx::_S_is_one_byte_char_type** (char *)
- template<class _CharT, class _Alloc>
  bool **__gnu_cxx::operator!=** (const _Rope_char_ptr_proxy< _CharT, _Alloc > &__x, const _Rope_char_ptr_↩proxy< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **__gnu_cxx::operator!=** (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_↩iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **__gnu_cxx::operator!=** (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **__gnu_cxx::operator!=** (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  _Rope_const_iterator< _CharT, _Alloc > **__gnu_cxx::operator+** (const _Rope_const_iterator< _CharT, _Alloc > &__x, std::ptrdiff_t __n)
- template<class _CharT, class _Alloc>
  _Rope_iterator< _CharT, _Alloc > **__gnu_cxx::operator+** (const _Rope_iterator< _CharT, _Alloc > &__x, std↩::ptrdiff_t __n)
- template<class _CharT, class _Alloc>
  rope< _CharT, _Alloc > **__gnu_cxx::operator+** (const rope< _CharT, _Alloc > &__left, _CharT __right)
- template<class _CharT, class _Alloc>
  rope< _CharT, _Alloc > **__gnu_cxx::operator+** (const rope< _CharT, _Alloc > &__left, const _CharT *__right)
- template<class _CharT, class _Alloc>
  rope< _CharT, _Alloc > **__gnu_cxx::operator+** (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)

- template<class _CharT, class _Alloc>
  _Rope_const_iterator< _CharT, _Alloc > **__gnu_cxx::operator+** (std::ptrdiff_t __n, const _Rope_const_iterator< _CharT, _Alloc > &__x)
- template<class _CharT, class _Alloc>
  _Rope_iterator< _CharT, _Alloc > **__gnu_cxx::operator+** (std::ptrdiff_t __n, const _Rope_iterator< _CharT, _Alloc > &__x)
- template<class _CharT, class _Alloc>
  rope< _CharT, _Alloc > & **__gnu_cxx::operator+=** (rope< _CharT, _Alloc > &__left, _CharT __right)
- template<class _CharT, class _Alloc>
  rope< _CharT, _Alloc > & **__gnu_cxx::operator+=** (rope< _CharT, _Alloc > &__left, const _CharT *__right)
- template<class _CharT, class _Alloc>
  rope< _CharT, _Alloc > & **__gnu_cxx::operator+=** (rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)
- template<class _CharT, class _Alloc>
  std::ptrdiff_t **__gnu_cxx::operator-** (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  _Rope_const_iterator< _CharT, _Alloc > **__gnu_cxx::operator-** (const _Rope_const_iterator< _CharT, _Alloc > &__x, std::ptrdiff_t __n)
- template<class _CharT, class _Alloc>
  std::ptrdiff_t **__gnu_cxx::operator-** (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  _Rope_iterator< _CharT, _Alloc > **__gnu_cxx::operator-** (const _Rope_iterator< _CharT, _Alloc > &__x, std::ptrdiff_t __n)
- template<class _CharT, class _Alloc>
  bool **__gnu_cxx::operator<** (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **__gnu_cxx::operator<** (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **__gnu_cxx::operator<** (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)
- template<class _CharT, class _Traits, class _Alloc>
  std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &__o, const rope< _CharT, _Alloc > &__r)
- template<class _CharT, class _Alloc>
  bool **__gnu_cxx::operator<=** (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **__gnu_cxx::operator<=** (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **__gnu_cxx::operator<=** (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **__gnu_cxx::operator==** (const _Rope_char_ptr_proxy< _CharT, _Alloc > &__x, const _Rope_char_ptr_proxy< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **__gnu_cxx::operator==** (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **__gnu_cxx::operator==** (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)

- template<class _CharT, class _Alloc>
  bool **__gnu_cxx::operator==** (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)
- template<class _CharT, class _Alloc>
  bool **__gnu_cxx::operator>** (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **__gnu_cxx::operator>** (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **__gnu_cxx::operator>** (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **__gnu_cxx::operator>=** (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **__gnu_cxx::operator>=** (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **__gnu_cxx::operator>=** (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)
- template<class _CharT, class __Alloc>
  void **__gnu_cxx::swap** (_Rope_char_ref_proxy< _CharT, __Alloc > __a, _Rope_char_ref_proxy< _CharT, _Alloc > __b)
- template<class _CharT, class _Alloc>
  void **__gnu_cxx::swap** (rope< _CharT, _Alloc > &__x, rope< _CharT, _Alloc > &__y)

Variables

- template<class _CharT, class _Alloc>
  rope< _CharT, _Alloc > **__gnu_cxx::identity_element** (_Rope_Concat_fn< _CharT, _Alloc >)
- template<class _CharT, class _Alloc>
  const rope< _CharT, _Alloc >::size_type **__gnu_cxx::rope< _CharT, _Alloc >::npos**

6.540.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

6.541 ropeimpl.h File Reference

Namespaces

- namespace __gnu_cxx

Macros

- #define **__GLIBCXX__**
- #define **__glibcxx_assert**(cond)
- #define **__N**(msgid)
- #define **__NO_CTYPE**
- #define **_GLIBCXX11_DEPRECATED**
- #define **_GLIBCXX11_DEPRECATED_SUGGEST**(ALT)
- #define **_GLIBCXX11_USE_C99_MATH**
- #define **_GLIBCXX11_USE_C99_STDIO**
- #define **_GLIBCXX11_USE_C99_STDLIB**
- #define **_GLIBCXX11_USE_C99_WCHAR**
- #define **_GLIBCXX14_DEPRECATED**

- #define **_GLIBCXX14_DEPRECATED_SUGGEST**(ALT)
- #define **_GLIBCXX17_DEPRECATED**
- #define **_GLIBCXX17_DEPRECATED_SUGGEST**(ALT)
- #define **_GLIBCXX20_DEPRECATED**
- #define **_GLIBCXX20_DEPRECATED_SUGGEST**(ALT)
- #define **_GLIBCXX23_DEPRECATED**
- #define **_GLIBCXX23_DEPRECATED_SUGGEST**(ALT)
- #define **_GLIBCXX26_DEPRECATED**
- #define **_GLIBCXX26_DEPRECATED_SUGGEST**(ALT)
- #define **_GLIBCXX98_USE_C99_COMPLEX**
- #define **_GLIBCXX98_USE_C99_MATH**
- #define **_GLIBCXX98_USE_C99_STDIO**
- #define **_GLIBCXX98_USE_C99_STDLIB**
- #define **_GLIBCXX98_USE_C99_WCHAR**
- #define **_GLIBCXX_ASSERT_FAIL**(_Condition)
- #define **_GLIBCXX_ATOMIC_WORD_BUILTINS**
- #define **_GLIBCXX_AUTO_CAST**(X)
- #define **_GLIBCXX_BEGIN_EXTERN_C**
- #define **_GLIBCXX_BEGIN_INLINE_ABI_NAMESPACE**(X)
- #define **_GLIBCXX_BEGIN_NAMESPACE_ALGO**
- #define **_GLIBCXX_BEGIN_NAMESPACE_CONTAINER**
- #define **_GLIBCXX_BEGIN_NAMESPACE_CXX11**
- #define **_GLIBCXX_BEGIN_NAMESPACE_LDBL**
- #define **_GLIBCXX_BEGIN_NAMESPACE_LDBL_OR_CXX11**
- #define **_GLIBCXX_BEGIN_NAMESPACE_VERSION**
- #define **_GLIBCXX_CAN_ALIGNAS_DESTRUCTIVE_SIZE**
- #define **_GLIBCXX_CPU_DEFINES**
- #define **_GLIBCXX_CXX_CONFIG_H**
- #define **_GLIBCXX_DEFAULT_ABI_TAG**
- #define **_GLIBCXX_DEPRECATED**
- #define **_GLIBCXX_DEPRECATED_SUGGEST**(ALT)
- #define **_GLIBCXX_DOXYGEN_ONLY**(X)
- #define **_GLIBCXX_END_EXTERN_C**
- #define **_GLIBCXX_END_INLINE_ABI_NAMESPACE**(X)
- #define **_GLIBCXX_END_NAMESPACE_ALGO**
- #define **_GLIBCXX_END_NAMESPACE_CONTAINER**
- #define **_GLIBCXX_END_NAMESPACE_CXX11**
- #define **_GLIBCXX_END_NAMESPACE_LDBL**
- #define **_GLIBCXX_END_NAMESPACE_LDBL_OR_CXX11**
- #define **_GLIBCXX_END_NAMESPACE_VERSION**
- #define **_GLIBCXX_EXTERN_TEMPLATE**
- #define **_GLIBCXX_EXTERN_TEMPLATE**
- #define **_GLIBCXX_FAST_MATH**
- #define **_GLIBCXX_FULLY_DYNAMIC_STRING**
- #define **_GLIBCXX_HAS_BUILTIN**(B)
- #define **_GLIBCXX_HAVE___CXA_THREAD_ATEXIT_IMPL**
- #define **_GLIBCXX_HAVE_ACOSF**
- #define **_GLIBCXX_HAVE_ACOSL**
- #define **_GLIBCXX_HAVE_ALIGNED_ALLOC**
- #define **_GLIBCXX_HAVE_ARC4RANDOM**
- #define **_GLIBCXX_HAVE_ARPA_INET_H**

- #define **_GLIBCXX_HAVE_AS_SYMVER_DIRECTIVE**
- #define **_GLIBCXX_HAVE_ASINF**
- #define **_GLIBCXX_HAVE_ASINL**
- #define **_GLIBCXX_HAVE_AT_QUICK_EXIT**
- #define **_GLIBCXX_HAVE_ATAN2F**
- #define **_GLIBCXX_HAVE_ATAN2L**
- #define **_GLIBCXX_HAVE_ATANF**
- #define **_GLIBCXX_HAVE_ATANL**
- #define **_GLIBCXX_HAVE_ATOMIC_LOCK_POLICY**
- #define **_GLIBCXX_HAVE_ATTRIBUTE_VISIBILITY**
- #define **_GLIBCXX_HAVE_BUILTIN_HAS_UNIQ_OBJ_REP**
- #define **_GLIBCXX_HAVE_BUILTIN_IS_AGGREGATE**
- #define **_GLIBCXX_HAVE_BUILTIN_LAUNDER**
- #define **_GLIBCXX_HAVE_C99_FLT_EVAL_TYPES**
- #define **_GLIBCXX_HAVE_CEILF**
- #define **_GLIBCXX_HAVE_CEILL**
- #define **_GLIBCXX_HAVE_COMPLEX_H**
- #define **_GLIBCXX_HAVE_COSF**
- #define **_GLIBCXX_HAVE_COSHF**
- #define **_GLIBCXX_HAVE_COSHL**
- #define **_GLIBCXX_HAVE_COSL**
- #define **_GLIBCXX_HAVE_DECL_STRNLEN**
- #define **_GLIBCXX_HAVE_DIRENT_H**
- #define **_GLIBCXX_HAVE_DIRFD**
- #define **_GLIBCXX_HAVE_DLFCN_H**
- #define **_GLIBCXX_HAVE_ENDIAN_H**
- #define **_GLIBCXX_HAVE_EXCEPTION_PTR_SINCE_GCC46**
- #define **_GLIBCXX_HAVE_EXECINFO_H**
- #define **_GLIBCXX_HAVE_EXPF**
- #define **_GLIBCXX_HAVE_EXPL**
- #define **_GLIBCXX_HAVE_FABSF**
- #define **_GLIBCXX_HAVE_FABSL**
- #define **_GLIBCXX_HAVE_FCNTL_H**
- #define **_GLIBCXX_HAVE_FDOPENDIR**
- #define **_GLIBCXX_HAVE_FENV_H**
- #define **_GLIBCXX_HAVE_FINITE**
- #define **_GLIBCXX_HAVE_FINITEF**
- #define **_GLIBCXX_HAVE_FINITEL**
- #define **_GLIBCXX_HAVE_FLOAT_H**
- #define **_GLIBCXX_HAVE_FLOORF**
- #define **_GLIBCXX_HAVE_FLOORL**
- #define **_GLIBCXX_HAVE_FMODF**
- #define **_GLIBCXX_HAVE_FMODL**
- #define **_GLIBCXX_HAVE_FREXPF**
- #define **_GLIBCXX_HAVE_FREXPL**
- #define **_GLIBCXX_HAVE_GETENTROPY**
- #define **_GLIBCXX_HAVE_GETIPINFO**
- #define **_GLIBCXX_HAVE_GETS**
- #define **_GLIBCXX_HAVE_HYPOT**
- #define **_GLIBCXX_HAVE_HYPOTF**
- #define **_GLIBCXX_HAVE_HYPOTL**

- #define **_GLIBCXX_HAVE_ICONV**
- #define **_GLIBCXX_HAVE_INTTYPES_H**
- #define **_GLIBCXX_HAVE_IS_CONSTANT_EVALUATED**
- #define **_GLIBCXX_HAVE_ISINF**
- #define **_GLIBCXX_HAVE_ISINFF**
- #define **_GLIBCXX_HAVE_ISINFL**
- #define **_GLIBCXX_HAVE_ISNAN**
- #define **_GLIBCXX_HAVE_ISNANF**
- #define **_GLIBCXX_HAVE_ISNANL**
- #define **_GLIBCXX_HAVE_ISWBLANK**
- #define **_GLIBCXX_HAVE_LC_MESSAGES**
- #define **_GLIBCXX_HAVE_LDEXPF**
- #define **_GLIBCXX_HAVE_LDEXPL**
- #define **_GLIBCXX_HAVE_LIBINTL_H**
- #define **_GLIBCXX_HAVE_LIMIT_AS**
- #define **_GLIBCXX_HAVE_LIMIT_DATA**
- #define **_GLIBCXX_HAVE_LIMIT_FSIZE**
- #define **_GLIBCXX_HAVE_LIMIT_RSS**
- #define **_GLIBCXX_HAVE_LIMIT_VMEM**
- #define **_GLIBCXX_HAVE_LINK**
- #define **_GLIBCXX_HAVE_LINK_H**
- #define **_GLIBCXX_HAVE_LINUX_FUTEX**
- #define **_GLIBCXX_HAVE_LINUX_RANDOM_H**
- #define **_GLIBCXX_HAVE_LINUX_TYPES_H**
- #define **_GLIBCXX_HAVE_LOCALE_H**
- #define **_GLIBCXX_HAVE_LOG10F**
- #define **_GLIBCXX_HAVE_LOG10L**
- #define **_GLIBCXX_HAVE_LOGF**
- #define **_GLIBCXX_HAVE_LOGL**
- #define **_GLIBCXX_HAVE_LSEEK**
- #define **_GLIBCXX_HAVE_MBSTATE_T**
- #define **_GLIBCXX_HAVE_MEMALIGN**
- #define **_GLIBCXX_HAVE_MEMORY_H**
- #define **_GLIBCXX_HAVE_MODFF**
- #define **_GLIBCXX_HAVE_MODFL**
- #define **_GLIBCXX_HAVE_NETDB_H**
- #define **_GLIBCXX_HAVE_NETINET_IN_H**
- #define **_GLIBCXX_HAVE_NETINET_TCP_H**
- #define **_GLIBCXX_HAVE_O_NONBLOCK**
- #define **_GLIBCXX_HAVE_OPENAT**
- #define **_GLIBCXX_HAVE_POLL**
- #define **_GLIBCXX_HAVE_POLL_H**
- #define **_GLIBCXX_HAVE_POSIX_MEMALIGN**
- #define **_GLIBCXX_HAVE_POWF**
- #define **_GLIBCXX_HAVE_POWL**
- #define **_GLIBCXX_HAVE_QUICK_EXIT**
- #define **_GLIBCXX_HAVE_READLINK**
- #define **_GLIBCXX_HAVE_S_ISREG**
- #define **_GLIBCXX_HAVE_SECURE_GETENV**
- #define **_GLIBCXX_HAVE_SETENV**
- #define **_GLIBCXX_HAVE_SINCOS**

- #define **_GLIBCXX_HAVE_SINCOSF**
- #define **_GLIBCXX_HAVE_SINCOSL**
- #define **_GLIBCXX_HAVE_SINF**
- #define **_GLIBCXX_HAVE_SINHF**
- #define **_GLIBCXX_HAVE_SINHL**
- #define **_GLIBCXX_HAVE_SINL**
- #define **_GLIBCXX_HAVE_SOCKATMARK**
- #define **_GLIBCXX_HAVE_SQRTF**
- #define **_GLIBCXX_HAVE_SQRTL**
- #define **_GLIBCXX_HAVE_STACKTRACE**
- #define **_GLIBCXX_HAVE_STDALIGN_H**
- #define **_GLIBCXX_HAVE_STDBOOL_H**
- #define **_GLIBCXX_HAVE_STDINT_H**
- #define **_GLIBCXX_HAVE_STDLIB_H**
- #define **_GLIBCXX_HAVE_STRERROR_L**
- #define **_GLIBCXX_HAVE_STRERROR_R**
- #define **_GLIBCXX_HAVE_STRING_H**
- #define **_GLIBCXX_HAVE_STRINGS_H**
- #define **_GLIBCXX_HAVE_STRTOF**
- #define **_GLIBCXX_HAVE_STRTOLD**
- #define **_GLIBCXX_HAVE_STRUCT_DIRENT_D_TYPE**
- #define **_GLIBCXX_HAVE_STRXFRM_L**
- #define **_GLIBCXX_HAVE_SYMLINK**
- #define **_GLIBCXX_HAVE_SYMVER_SYMBOL_RENAMING_RUNTIME_SUPPORT**
- #define **_GLIBCXX_HAVE_SYS_IOCTL_H**
- #define **_GLIBCXX_HAVE_SYS_IPC_H**
- #define **_GLIBCXX_HAVE_SYS_MMAN_H**
- #define **_GLIBCXX_HAVE_SYS_PARAM_H**
- #define **_GLIBCXX_HAVE_SYS_PTRACE_H**
- #define **_GLIBCXX_HAVE_SYS_RESOURCE_H**
- #define **_GLIBCXX_HAVE_SYS_SEM_H**
- #define **_GLIBCXX_HAVE_SYS_SOCKET_H**
- #define **_GLIBCXX_HAVE_SYS_STAT_H**
- #define **_GLIBCXX_HAVE_SYS_STATVFS_H**
- #define **_GLIBCXX_HAVE_SYS_SYSINFO_H**
- #define **_GLIBCXX_HAVE_SYS_TIME_H**
- #define **_GLIBCXX_HAVE_SYS_TYPES_H**
- #define **_GLIBCXX_HAVE_SYS_UIO_H**
- #define **_GLIBCXX_HAVE_TANF**
- #define **_GLIBCXX_HAVE_TANHF**
- #define **_GLIBCXX_HAVE_TANHL**
- #define **_GLIBCXX_HAVE_TANL**
- #define **_GLIBCXX_HAVE_TGMATH_H**
- #define **_GLIBCXX_HAVE_TIMESPEC_GET**
- #define **_GLIBCXX_HAVE_TLS**
- #define **_GLIBCXX_HAVE_TRUNCATE**
- #define **_GLIBCXX_HAVE_UCHAR_H**
- #define **_GLIBCXX_HAVE_UNISTD_H**
- #define **_GLIBCXX_HAVE_UNLINKAT**
- #define **_GLIBCXX_HAVE_USELOCALE**
- #define **_GLIBCXX_HAVE_UTIME_H**

- #define **_GLIBCXX_HAVE_VFWSCANF**
- #define **_GLIBCXX_HAVE_VSWSCANF**
- #define **_GLIBCXX_HAVE_VWSCANF**
- #define **_GLIBCXX_HAVE_WCHAR_H**
- #define **_GLIBCXX_HAVE_WCSTOF**
- #define **_GLIBCXX_HAVE_WCTYPE_H**
- #define **_GLIBCXX_HAVE_WRITEV**
- #define **_GLIBCXX_HOSTED**
- #define **_GLIBCXX_ICONV_CONST**
- #define **_GLIBCXX_INLINE_VERSION**
- #define **_GLIBCXX_LT_OBJDIR**
- #define **_GLIBCXX_MANGLE_SIZE_T**
- #define **_GLIBCXX_MAY_HAVE___CXA_THREAD_ATEXIT_IMPL**
- #define **_GLIBCXX_NAMESPACE_CXX11**
- #define **_GLIBCXX_NAMESPACE_LDBL**
- #define **_GLIBCXX_NAMESPACE_LDBL_OR_CXX11**
- #define **_GLIBCXX_NO_OBSOLETE_ISINF_ISNAN_DYNAMIC**
- #define **_GLIBCXX_NODISCARD**
- #define **_GLIBCXX_NOEXCEPT_PARM**
- #define **_GLIBCXX_NOEXCEPT_QUAL**
- #define **_GLIBCXX_OS_DEFINES**
- #define **_GLIBCXX_PARALLEL_FEATURES_H**
- #define **_GLIBCXX_RELEASE**
- #define **_GLIBCXX_RES_LIMITS**
- #define **_GLIBCXX_STATIC_TZDATA**
- #define **_GLIBCXX_STD_A**
- #define **_GLIBCXX_STD_C**
- #define **_GLIBCXX_STDIO_EOF**
- #define **_GLIBCXX_STDIO_SEEK_CUR**
- #define **_GLIBCXX_STDIO_SEEK_END**
- #define **_GLIBCXX_SYMVER**
- #define **_GLIBCXX_SYMVER_GNU**
- #define **_GLIBCXX_TXN_SAFE**
- #define **_GLIBCXX_TXN_SAFE_DYN**
- #define **_GLIBCXX_USE_ALLOCATOR_NEW**
- #define **_GLIBCXX_USE_BUILTIN_TRAIT**(BT)
- #define **_GLIBCXX_USE_C11_UCHAR_CXX11**
- #define **_GLIBCXX_USE_C99**
- #define **_GLIBCXX_USE_C99_COMPLEX_TR1**
- #define **_GLIBCXX_USE_C99_CTYPE**
- #define **_GLIBCXX_USE_C99_CTYPE_TR1**
- #define **_GLIBCXX_USE_C99_FENV**
- #define **_GLIBCXX_USE_C99_FENV_TR1**
- #define **_GLIBCXX_USE_C99_INTTYPES**
- #define **_GLIBCXX_USE_C99_INTTYPES_TR1**
- #define **_GLIBCXX_USE_C99_INTTYPES_WCHAR_T**
- #define **_GLIBCXX_USE_C99_INTTYPES_WCHAR_T_TR1**
- #define **_GLIBCXX_USE_C99_MATH_FUNCS**
- #define **_GLIBCXX_USE_C99_MATH_TR1**
- #define **_GLIBCXX_USE_C99_STDINT**
- #define **_GLIBCXX_USE_C99_STDINT_TR1**

- #define **_GLIBCXX_USE_CHDIR**
- #define **_GLIBCXX_USE_CHMOD**
- #define **_GLIBCXX_USE_CLOCK_MONOTONIC**
- #define **_GLIBCXX_USE_CLOCK_REALTIME**
- #define **_GLIBCXX_USE_DECIMAL_FLOAT**
- #define **_GLIBCXX_USE_DEV_RANDOM**
- #define **_GLIBCXX_USE_DUAL_ABI**
- #define **_GLIBCXX_USE_FCHMOD**
- #define **_GLIBCXX_USE_FCHMODAT**
- #define **_GLIBCXX_USE_FSEEKO_FTELLO**
- #define **_GLIBCXX_USE_GET_NPROCS**
- #define **_GLIBCXX_USE_GETCWD**
- #define **_GLIBCXX_USE_GETTIMEOFDAY**
- #define **_GLIBCXX_USE_INIT_PRIORITY_ATTRIBUTE**
- #define **_GLIBCXX_USE_LFS**
- #define **_GLIBCXX_USE_LONG_LONG**
- #define **_GLIBCXX_USE_LSTAT**
- #define **_GLIBCXX_USE_MKDIR**
- #define **_GLIBCXX_USE_NANOSLEEP**
- #define **_GLIBCXX_USE_NL_LANGINFO_L**
- #define **_GLIBCXX_USE_NLS**
- #define **_GLIBCXX_USE_PROC_SELF_STATUS**
- #define **_GLIBCXX_USE_PTHREAD_COND_CLOCKWAIT**
- #define **_GLIBCXX_USE_PTHREAD_MUTEX_CLOCKLOCK**
- #define **_GLIBCXX_USE_PTHREAD_RWLOCK_CLOCKLOCK**
- #define **_GLIBCXX_USE_RANDOM_TR1**
- #define **_GLIBCXX_USE_REALPATH**
- #define **_GLIBCXX_USE_SC_NPROCESSORS_ONLN**
- #define **_GLIBCXX_USE_SCHED_YIELD**
- #define **_GLIBCXX_USE_SENDFILE**
- #define **_GLIBCXX_USE_ST_MTIM**
- #define **_GLIBCXX_USE_STD_SPEC_FUNCS**
- #define **_GLIBCXX_USE_STRUCT_TM_TM_ZONE**
- #define **_GLIBCXX_USE_TMPNAM**
- #define **_GLIBCXX_USE_UCHAR_C8RTOMB_MBRTOC8_CXX20**
- #define **_GLIBCXX_USE_UCHAR_C8RTOMB_MBRTOC8_FCHAR8_T**
- #define **_GLIBCXX_USE_UTIME**
- #define **_GLIBCXX_USE_UTIMENSAT**
- #define **_GLIBCXX_USE_WCHAR_T**
- #define **_GLIBCXX_VERBOSE**
- #define **_GLIBCXX_VERBOSE_ASSERT**
- #define **_GLIBCXX_VISIBILITY**(V)
- #define **_GLIBCXX_X86_RDRAND**
- #define **_GLIBCXX_X86_RDSEED**
- #define **_GLIBCXX_ZONEINFO_DIR**
- #define **_GTHREAD_USE_MUTEX_TIMEDLOCK**
- #define **_REQUIRES_FREESTANDING_H**

Functions

- template<class _CharT, class _Traits>
  void **__gnu_cxx::_Rope_fill** (std::basic_ostream< _CharT, _Traits > &__o, std::size_t __n)
- template<class _CharT>
  bool **__gnu_cxx::_Rope_is_simple** (_CharT *)
- bool **__gnu_cxx::_Rope_is_simple** (char *)
- bool **__gnu_cxx::_Rope_is_simple** (wchar_t *)
- template<class _Rope_iterator>
  void **__gnu_cxx::_Rope_rotate** (_Rope_iterator __first, _Rope_iterator __middle, _Rope_iterator __last)
- template<class _CharT, class _Traits, class _Alloc>
  std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator**<< (std::basic_ostream< _CharT, _Traits > &↩
  __o, const rope< _CharT, _Alloc > &__r)
- void **__gnu_cxx::rotate** (_Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __first, _Rope_↩
  iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __middle, _Rope_iterator< char, __STL_DEFAULT_↩
  ALLOCATOR(char)> __last)

Variables

- template<class _CharT, class _Alloc>
  _CharT **__gnu_cxx::rope< _CharT, _Alloc >::_S_empty_c_str** [1]
- template<class _CharT, class _Alloc>
  const unsigned long **__gnu_cxx::rope< _CharT, _Alloc >::_S_min_len** [int(__detail::_S_max_rope_depth)+1]

6.541.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ext/rope>.

6.542 slist File Reference

Classes

- class __gnu_cxx::slist< _Tp, _Alloc >

Namespaces

- namespace __gnu_cxx
- namespace std

Macros

- #define **_SLIST**

Functions

- _Slist_node_base * **__gnu_cxx::__slist_make_link** (_Slist_node_base *__prev_node, _Slist_node_base *_↩
  _new_node)
- _Slist_node_base * **__gnu_cxx::__slist_previous** (_Slist_node_base *__head, const _Slist_node_base *__↩
  node)
- const _Slist_node_base * **__gnu_cxx::__slist_previous** (const _Slist_node_base *__head, const _Slist_↩
  node_base *__node)
- _Slist_node_base * **__gnu_cxx::__slist_reverse** (_Slist_node_base *__node)
- std::size_t **__gnu_cxx::__slist_size** (_Slist_node_base *__node)

- void **__gnu_cxx::__slist_splice_after** (_Slist_node_base *__pos, _Slist_node_base *__before_first, _Slist_↩ node_base *__before_last)
- void **__gnu_cxx::__slist_splice_after** (_Slist_node_base *__pos, _Slist_node_base *__head)
- template<class _Tp, class _Alloc>
  bool **__gnu_cxx::operator!=** (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)
- template<class _Tp, class _Alloc>
  bool **__gnu_cxx::operator<** (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)
- template<class _Tp, class _Alloc>
  bool **__gnu_cxx::operator<=** (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)
- template<class _Tp, class _Alloc>
  bool **__gnu_cxx::operator==** (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)
- template<class _Tp, class _Alloc>
  bool **__gnu_cxx::operator>** (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)
- template<class _Tp, class _Alloc>
  bool **__gnu_cxx::operator>=** (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)
- template<class _Tp, class _Alloc>
  void **__gnu_cxx::swap** (slist< _Tp, _Alloc > &__x, slist< _Tp, _Alloc > &__y)

6.542.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

6.543 sso_string_base.h File Reference

Namespaces

- namespace __gnu_cxx

6.543.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ext/vstring.h>.

6.544 stdio_filebuf.h File Reference

Classes

- class __gnu_cxx::stdio_filebuf< _CharT, _Traits >

Namespaces

- namespace __gnu_cxx

6.544.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

6.545 stdio_sync_filebuf.h File Reference

Classes

- class __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >

Namespaces

- namespace __gnu_cxx

6.545.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

6.546 string_conversions.h File Reference

Namespaces

- namespace __gnu_cxx

Functions

- template<typename _TRet, typename _Ret = _TRet, typename _CharT, typename... _Base>
  _Ret **__gnu_cxx::__stoa** (_TRet(*__convf)(const _CharT *, _CharT **, _Base...), const char *__name, const _CharT *__str, std::size_t *__idx, _Base... __base)
- template<typename _String, typename _CharT = typename _String::value_type>
  _String **__gnu_cxx::__to_xstring** (int(*__convf)(_CharT *, std::size_t, const _CharT *, __builtin_va_list), std::size_t __n, const _CharT *__fmt,...)

6.546.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

6.547 throw_allocator.h File Reference

Classes

- struct __gnu_cxx::limit_condition::always_adjustor
- struct __gnu_cxx::random_condition::always_adjustor
- struct __gnu_cxx::annotate_base
- struct __gnu_cxx::condition_base
- struct __gnu_cxx::forced_error
- struct __gnu_cxx::random_condition::group_adjustor
- struct std::hash< __gnu_cxx::throw_value_limit >
- struct std::hash< __gnu_cxx::throw_value_random >
- struct __gnu_cxx::limit_condition::limit_adjustor
- struct __gnu_cxx::limit_condition
- struct __gnu_cxx::limit_condition::never_adjustor
- struct __gnu_cxx::random_condition::never_adjustor
- struct __gnu_cxx::random_condition
- class __gnu_cxx::throw_allocator_base< _Tp, _Cond >
- struct __gnu_cxx::throw_allocator_limit< _Tp >
- struct __gnu_cxx::throw_allocator_random< _Tp >
- struct __gnu_cxx::throw_value_base< _Cond >
- struct __gnu_cxx::throw_value_limit
- struct __gnu_cxx::throw_value_random

Namespaces

- namespace __gnu_cxx
- namespace std

Functions

- void **__gnu_cxx::__throw_forced_error** ()
- template<typename _Cond>
  throw_value_base< _Cond > **__gnu_cxx::operator*** (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- template<typename _Cond>
  throw_value_base< _Cond > **__gnu_cxx::operator+** (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- template<typename _Cond>
  throw_value_base< _Cond > **__gnu_cxx::operator-** (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- template<typename _Cond>
  bool **__gnu_cxx::operator<** (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- std::ostream & **__gnu_cxx::operator<<** (std::ostream &os, const annotate_base &__b)
- template<typename _Tp, typename _Cond>
  bool **__gnu_cxx::operator==** (const throw_allocator_base< _Tp, _Cond > &, const throw_allocator_base< _Tp, _Cond > &)
- template<typename _Cond>
  bool **__gnu_cxx::operator==** (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- template<typename _Cond>
  void **__gnu_cxx::swap** (throw_value_base< _Cond > &__a, throw_value_base< _Cond > &__b)

6.547.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.
Contains two exception-generating types (throw_value, throw_allocator) intended to be used as value and allocator types while testing exception safety in templatized containers and algorithms. The allocator has additional log and debug features. The exception generated is of type forced_exception_error.

6.548 type_traits.h File Reference

Namespaces

- namespace __gnu_cxx

Functions

- template<typename _Type>
  constexpr bool **__gnu_cxx::__is_null_pointer** (_Type *__ptr)
- template<typename _Type>
  constexpr bool **__gnu_cxx::__is_null_pointer** (_Type)
- constexpr bool **__gnu_cxx::__is_null_pointer** (std::nullptr_t)

6.548.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

6.549 typelist.h File Reference

Namespaces

- namespace __gnu_cxx
- namespace __gnu_cxx::typelist

Macros

- #define **_GLIBCXX_TYPELIST_CHAIN1**(X0)
- #define **_GLIBCXX_TYPELIST_CHAIN10**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9)
- #define **_GLIBCXX_TYPELIST_CHAIN11**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10)
- #define **_GLIBCXX_TYPELIST_CHAIN12**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11)
- #define **_GLIBCXX_TYPELIST_CHAIN13**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12)
- #define **_GLIBCXX_TYPELIST_CHAIN14**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13)
- #define **_GLIBCXX_TYPELIST_CHAIN15**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14)
- #define **_GLIBCXX_TYPELIST_CHAIN16**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15)
- #define **_GLIBCXX_TYPELIST_CHAIN17**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16)
- #define **_GLIBCXX_TYPELIST_CHAIN18**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X17)
- #define **_GLIBCXX_TYPELIST_CHAIN19**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X17, X18)
- #define **_GLIBCXX_TYPELIST_CHAIN2**(X0, X1)
- #define **_GLIBCXX_TYPELIST_CHAIN20**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X17, X18, X19)
- #define **_GLIBCXX_TYPELIST_CHAIN3**(X0, X1, X2)
- #define **_GLIBCXX_TYPELIST_CHAIN4**(X0, X1, X2, X3)
- #define **_GLIBCXX_TYPELIST_CHAIN5**(X0, X1, X2, X3, X4)
- #define **_GLIBCXX_TYPELIST_CHAIN6**(X0, X1, X2, X3, X4, X5)
- #define **_GLIBCXX_TYPELIST_CHAIN7**(X0, X1, X2, X3, X4, X5, X6)
- #define **_GLIBCXX_TYPELIST_CHAIN8**(X0, X1, X2, X3, X4, X5, X6, X7)
- #define **_GLIBCXX_TYPELIST_CHAIN9**(X0, X1, X2, X3, X4, X5, X6, X7, X8)

Functions

- template<typename Fn, typename Typelist>
  void **__gnu_cxx::typelist::apply** (Fn &, Typelist)
- template<typename Fn, typename Typelist>
  void **__gnu_cxx::typelist::apply_generator** (Fn &fn, Typelist)
- template<typename Fn, typename TypelistT, typename TypelistV>
  void **__gnu_cxx::typelist::apply_generator** (Fn &fn, TypelistT, TypelistV)
- template<typename Gn, typename Typelist>
  void __gnu_cxx::typelist::apply_generator (Gn &, Typelist)
- template<typename Gn, typename TypelistT, typename TypelistV>
  void **__gnu_cxx::typelist::apply_generator** (Gn &, TypelistT, TypelistV)

6.549.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.
Contains typelist_chain definitions. Typelists are an idea by Andrei Alexandrescu.

6.550 vstring.h File Reference

Classes

- class __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >
- struct std::hash< __gnu_cxx::__u16vstring >
- struct std::hash< __gnu_cxx::__u32vstring >
- struct std::hash< __gnu_cxx::__vstring >
- struct std::hash< __gnu_cxx::__wvstring >

Namespaces

- namespace __gnu_cxx
- namespace std

Functions

- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str, _CharT __delim)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator!= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator!= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *_↩ _rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator!= (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &_↩ _rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base > **__gnu_cxx::operator+** (__versa_string< _CharT, _Traits, ↩ _Alloc, _Base > &&__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &&__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base > **__gnu_cxx::operator+** (__versa_string< _CharT, _Traits, ↩ _Alloc, _Base > &&__lhs, _CharT __rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base > **__gnu_cxx::operator+** (__versa_string< _CharT, _Traits, ↩ _Alloc, _Base > &&__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base > **__gnu_cxx::operator+** (__versa_string< _CharT, _Traits, ↩ _Alloc, _Base > &&__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base > **__gnu_cxx::operator+** (_CharT __lhs, __versa_string< _↩ CharT, _Traits, _Alloc, _Base > &&__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (_CharT __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base > **__gnu_cxx::operator+** (const __versa_string< _CharT, _↩ Traits, _Alloc, _Base > &__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &&__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, _CharT __rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)

- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base > **__gnu_cxx::operator+** (const _CharT *__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &&__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator< (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator< (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *_↩ _rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator< (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &_↩ _rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator<= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator<= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator<= (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator== (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator== (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *↩ __rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator== (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &↩ __rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator> (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator> (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *_↩ _rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator> (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &_↩ _rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator>= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator>= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> bool __gnu_cxx::operator>= (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)

- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str)
- double **__gnu_cxx::stod** (const __vstring &__str, std::size_t *__idx=0)
- double **__gnu_cxx::stod** (const __wvstring &__str, std::size_t *__idx=0)
- float **__gnu_cxx::stof** (const __vstring &__str, std::size_t *__idx=0)
- float **__gnu_cxx::stof** (const __wvstring &__str, std::size_t *__idx=0)
- int **__gnu_cxx::stoi** (const __vstring &__str, std::size_t *__idx=0, int __base=10)
- int **__gnu_cxx::stoi** (const __wvstring &__str, std::size_t *__idx=0, int __base=10)
- long **__gnu_cxx::stol** (const __vstring &__str, std::size_t *__idx=0, int __base=10)
- long **__gnu_cxx::stol** (const __wvstring &__str, std::size_t *__idx=0, int __base=10)
- long double **__gnu_cxx::stold** (const __vstring &__str, std::size_t *__idx=0)
- long double **__gnu_cxx::stold** (const __wvstring &__str, std::size_t *__idx=0)
- long long **__gnu_cxx::stoll** (const __vstring &__str, std::size_t *__idx=0, int __base=10)
- long long **__gnu_cxx::stoll** (const __wvstring &__str, std::size_t *__idx=0, int __base=10)
- unsigned long **__gnu_cxx::stoul** (const __vstring &__str, std::size_t *__idx=0, int __base=10)
- unsigned long **__gnu_cxx::stoul** (const __wvstring &__str, std::size_t *__idx=0, int __base=10)
- unsigned long long **__gnu_cxx::stoull** (const __vstring &__str, std::size_t *__idx, int __base=10)
- unsigned long long **__gnu_cxx::stoull** (const __wvstring &__str, std::size_t *__idx=0, int __base=10)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> void __gnu_cxx::swap (__versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, __versa_string< _CharT, _↩ Traits, _Alloc, _Base > &__rhs)
- __vstring **__gnu_cxx::to_string** (double __val)
- __vstring **__gnu_cxx::to_string** (float __val)
- __vstring **__gnu_cxx::to_string** (int __val)
- __vstring **__gnu_cxx::to_string** (long __val)
- __vstring **__gnu_cxx::to_string** (long double __val)
- __vstring **__gnu_cxx::to_string** (long long __val)
- __vstring **__gnu_cxx::to_string** (unsigned __val)
- __vstring **__gnu_cxx::to_string** (unsigned long __val)
- __vstring **__gnu_cxx::to_string** (unsigned long long __val)
- __wvstring **__gnu_cxx::to_wstring** (double __val)
- __wvstring **__gnu_cxx::to_wstring** (float __val)
- __wvstring **__gnu_cxx::to_wstring** (int __val)
- __wvstring **__gnu_cxx::to_wstring** (long __val)
- __wvstring **__gnu_cxx::to_wstring** (long double __val)
- __wvstring **__gnu_cxx::to_wstring** (long long __val)
- __wvstring **__gnu_cxx::to_wstring** (unsigned __val)
- __wvstring **__gnu_cxx::to_wstring** (unsigned long __val)
- __wvstring **__gnu_cxx::to_wstring** (unsigned long long __val)

6.550.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

6.551 vstring.tcc File Reference

Namespaces

- namespace __gnu_cxx
- namespace std

Macros

- #define **_VSTRING_TCC**

Functions

- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str, _CharT __delim)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (_CharT __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, _CharT __rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> __versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str)

Variables

- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type **__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::npos**

6.551.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ext/vstring.h>.

6.552 vstring_fwd.h File Reference

Namespaces

- namespace __gnu_cxx

Typedefs

- typedef __versa_string< char, std::char_traits< char >, std::allocator< char >, __rc_string_base > **__gnu_cxx::__rc_string**
- typedef __vstring **__gnu_cxx::__sso_string**
- typedef __versa_string< char16_t, std::char_traits< char16_t >, std::allocator< char16_t >, __rc_string_base > **__gnu_cxx::__u16rc_string**
- typedef __u16vstring **__gnu_cxx::__u16sso_string**
- typedef __versa_string< char16_t > **__gnu_cxx::__u16vstring**

- typedef __versa_string< char32_t, std::char_traits< char32_t >, std::allocator< char32_t >, __rc_string_base > **__gnu_cxx::__u32rc_string**
- typedef __u32vstring **__gnu_cxx::__u32sso_string**
- typedef __versa_string< char32_t > **__gnu_cxx::__u32vstring**
- typedef __versa_string< char > **__gnu_cxx::__vstring**
- typedef __versa_string< wchar_t, std::char_traits< wchar_t >, std::allocator< wchar_t >, __rc_string_base > **__gnu_cxx::__wrc_string**
- typedef __wvstring **__gnu_cxx::__wsso_string**
- typedef __versa_string< wchar_t > **__gnu_cxx::__wvstring**

6.552.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ext/vstring.h>.

6.553 vstring_util.h File Reference

Namespaces

- namespace __gnu_cxx

6.553.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ext/vstring.h>.

6.554 fenv.h File Reference

6.554.1 Detailed Description

This is a Standard C++ Library header.

6.555 experimental/filesystem File Reference

Macros

- #define **__cpp_lib_experimental_filesystem**
- #define **_GLIBCXX_EXPERIMENTAL_FILESYSTEM**

6.555.1 Detailed Description

This is a TS C++ Library header.

6.556 filesystem File Reference

Macros

- #define **__glibcxx_want_filesystem**
- #define **_GLIBCXX_FILESYSTEM**

6.556.1 Detailed Description

This is a Standard C++ Library header.

6.557 flat_map File Reference

Macros

- #define **__glibcxx_want_flat_map**
- #define **_GLIBCXX_FLAT_MAP**

6.557.1 Detailed Description

This is a Standard C++ Library header.

6.558 flat_set File Reference

Macros

- #define **__glibcxx_want_flat_set**
- #define **_GLIBCXX_FLAT_SET**

6.558.1 Detailed Description

This is a Standard C++ Library header.

6.559 format File Reference

Macros

- #define **__glibcxx_want_format**
- #define **__glibcxx_want_format_ranges**
- #define **__glibcxx_want_format_uchar**
- #define **_GLIBCXX_FORMAT**

6.559.1 Detailed Description

This is a Standard C++ Library header.

6.560 debug/forward_list File Reference

Classes

- class __gnu_debug::_Safe_forward_list< _SafeSequence >
- class std::__debug::forward_list< _Tp, _Alloc >

Namespaces

- namespace __gnu_debug
- namespace std
- namespace std::__debug

Macros

- #define **__glibcxx_check_valid_fl_range**(_First, _Last, _Dist)
- #define **_GLIBCXX20_ONLY**(__expr)
- #define **_GLIBCXX_DEBUG_FORWARD_LIST**
- #define **_GLIBCXX_FWDLIST_REMOVE_RETURN_TYPE_TAG**

Functions

- template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  std::__debug::forward_list (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> forward_list< _ValT, _Allocator >
- template<typename _Tp, typename _Allocator = allocator<_Tp>, typename = _RequireAllocator<_Allocator>>
  std::__debug::forward_list (size_t, _Tp, _Allocator=_Allocator()) -> forward_list< _Tp, _Allocator >
- template<typename _Tp, typename _Alloc>
  constexpr __detail::__synth3way_t< _Tp > **std::__debug::operator<=>** (const forward_list< _Tp, _Alloc > &__x, const forward_list< _Tp, _Alloc > &__y)
- template<typename _Tp, typename _Alloc>
  bool **std::__debug::operator==** (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)
- template<typename _Tp, typename _Alloc>
  void std::__debug::swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly) noexcept(noexcept(__lx.swap(__ly)))

6.560.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

6.561 experimental/forward_list File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **_GLIBCXX_EXPERIMENTAL_FORWARD_LIST**

Typedefs

- template<typename _Tp>
  using **std::experimental::fundamentals_v2::pmr::forward_list**

Functions

- template<typename _Tp, typename _Alloc, typename _Up>
  void **std::experimental::erase** (forward_list< _Tp, _Alloc > &__cont, const _Up &__value)
- template<typename _Tp, typename _Alloc, typename _Predicate>
  void **std::experimental::erase_if** (forward_list< _Tp, _Alloc > &__cont, _Predicate __pred)

6.561.1 Detailed Description

This is a TS C++ Library header.

6.562 forward_list File Reference

Namespaces

- namespace std

Macros

- #define **__glibcxx_want_algorithm_default_value_type**
- #define **__glibcxx_want_allocator_traits_is_always_equal**
- #define **__glibcxx_want_containers_ranges**
- #define **__glibcxx_want_erase_if**
- #define **__glibcxx_want_incomplete_container_elements**
- #define **__glibcxx_want_list_remove_return_type**
- #define **__glibcxx_want_nonmember_container_access**
- #define **_GLIBCXX_FORWARD_LIST**

Typedefs

- template<typename _Tp>
  using **std::pmr::forward_list**

6.562.1 Detailed Description

This is a Standard C++ Library header.

6.563 fstream File Reference

Classes

- class std::basic_filebuf< _CharT, _Traits >
- class std::basic_fstream< _CharT, _Traits >
- class std::basic_ifstream< _CharT, _Traits >
- class std::basic_ofstream< _CharT, _Traits >

Namespaces

- namespace std

Macros

- #define **__glibcxx_want_fstream_native_handle**
- #define **_GLIBCXX_BUFSIZ**
- #define **_GLIBCXX_FSTREAM**

Typedefs

- template<typename _Path, typename _Result = _Path, typename _Path2 = decltype(std::declval<_Path&>().make_preferred().filename())>
  using **std::_If_fs_path**

Functions

- template<class _CharT, class _Traits>
  void std::swap (basic_filebuf< _CharT, _Traits > &__x, basic_filebuf< _CharT, _Traits > &__y)
- template<class _CharT, class _Traits>
  void std::swap (basic_fstream< _CharT, _Traits > &__x, basic_fstream< _CharT, _Traits > &__y)
- template<class _CharT, class _Traits>
  void std::swap (basic_ifstream< _CharT, _Traits > &__x, basic_ifstream< _CharT, _Traits > &__y)
- template<class _CharT, class _Traits>
  void std::swap (basic_ofstream< _CharT, _Traits > &__x, basic_ofstream< _CharT, _Traits > &__y)

6.563.1 Detailed Description

This is a Standard C++ Library header.

6.564 experimental/functional File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **__cpp_lib_experimental_boyer_moore_searching**
- #define **__cpp_lib_experimental_not_fn**
- #define **_GLIBCXX_EXPERIMENTAL_FUNCTIONAL**

Typedefs

- template<typename _RAIter, typename _Hash, typename _Pred, typename _Val = typename iterator_traits<_RAIter>::value_type, typename _Diff = typename iterator_traits<_RAIter>::difference_type>
  using **std::experimental::__boyer_moore_base_t**

Functions

- template<typename _RAIter, typename _Hash = std::hash<typename std::iterator_traits<_RAIter>::value_type>, typename _BinaryPredicate = equal_to<>>
  boyer_moore_horspool_searcher< _RAIter, _Hash, _BinaryPredicate > std::experimental::make_boyer_moore_horspool_searcher (_RAIter __pat_first, _RAIter __pat_last, _Hash __hf=_Hash(), _BinaryPredicate __pred=_BinaryPredicate())
- template<typename _RAIter, typename _Hash = std::hash<typename std::iterator_traits<_RAIter>::value_type>, typename _BinaryPredicate = equal_to<>>
  boyer_moore_searcher< _RAIter, _Hash, _BinaryPredicate > std::experimental::make_boyer_moore_searcher (_RAIter __pat_first, _RAIter __pat_last, _Hash __hf=_Hash(), _BinaryPredicate __pred=_BinaryPredicate())
- template<typename _ForwardIterator, typename _BinaryPredicate = std::equal_to<>>
  default_searcher< _ForwardIterator, _BinaryPredicate > std::experimental::make_default_searcher (_ForwardIterator __pat_first, _ForwardIterator __pat_last, _BinaryPredicate __pred=_BinaryPredicate())
- template<typename _Fn>
  auto std::experimental::not_fn (_Fn &&__fn) noexcept(std::is_nothrow_constructible< std::decay_t< _Fn >, _Fn && >::value)

Variables

- template<typename _Tp>
  constexpr bool std::experimental::is_bind_expression_v
- template<typename _Tp>
  constexpr int std::experimental::is_placeholder_v

6.564.1 Detailed Description

This is a TS C++ Library header.

6.565 ext/functional File Reference

Classes

- class __gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >
- struct __gnu_cxx::constant_binary_fun< _Result, _Arg1, _Arg2 >
- struct __gnu_cxx::constant_unary_fun< _Result, _Argument >
- struct __gnu_cxx::constant_void_fun< _Result >
- struct __gnu_cxx::project1st< _Arg1, _Arg2 >
- struct __gnu_cxx::project2nd< _Arg1, _Arg2 >
- struct __gnu_cxx::select1st< _Pair >
- struct __gnu_cxx::select2nd< _Pair >
- class __gnu_cxx::subtractive_rng
- class __gnu_cxx::unary_compose< _Operation1, _Operation2 >

Namespaces

- namespace __gnu_cxx

Macros

- #define **_EXT_FUNCTIONAL**

Functions

- template<class _Operation1, class _Operation2>
  unary_compose< _Operation1, _Operation2 > __gnu_cxx::compose1 (const _Operation1 &__fn1, const _Operation2 &__fn2)
- template<class _Operation1, class _Operation2, class _Operation3>
  binary_compose< _Operation1, _Operation2, _Operation3 > __gnu_cxx::compose2 (const _Operation1 &__fn1, const _Operation2 &__fn2, const _Operation3 &__fn3)
- template<class _Result>
  constant_void_fun< _Result > __gnu_cxx::constant0 (const _Result &__val)
- template<class _Result>
  constant_unary_fun< _Result, _Result > __gnu_cxx::constant1 (const _Result &__val)
- template<class _Result>
  constant_binary_fun< _Result, _Result, _Result > __gnu_cxx::constant2 (const _Result &__val)
- template<class _Tp>
  _Tp __gnu_cxx::identity_element (std::multiplies< _Tp >)
- template<class _Tp>
  _Tp __gnu_cxx::identity_element (std::plus< _Tp >)
- template<class _Ret, class _Tp, class _Arg>
  std::const_mem_fun1_t< _Ret, _Tp, _Arg > **__gnu_cxx::mem_fun1** (_Ret(_Tp::*__f)(_Arg) const)
- template<class _Ret, class _Tp, class _Arg>
  std::mem_fun1_t< _Ret, _Tp, _Arg > **__gnu_cxx::mem_fun1** (_Ret(_Tp::*__f)(_Arg))
- template<class _Ret, class _Tp, class _Arg>
  std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg > **__gnu_cxx::mem_fun1_ref** (_Ret(_Tp::*__f)(_Arg) const)
- template<class _Ret, class _Tp, class _Arg>
  std::mem_fun1_ref_t< _Ret, _Tp, _Arg > **__gnu_cxx::mem_fun1_ref** (_Ret(_Tp::*__f)(_Arg))

6.565.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

6.566 functional File Reference

Classes

- class std::_Not_fn< _Fn >
- struct std::_Placeholder< _Num >
- struct std::is_bind_expression< _Tp >
- struct std::is_bind_expression< _Bind< _Signature > >
- struct std::is_bind_expression< _Bind_result< _Result, _Signature > >
- struct std::is_bind_expression< const _Bind< _Signature > >
- struct std::is_bind_expression< const _Bind_result< _Result, _Signature > >
- struct std::is_bind_expression< const volatile _Bind< _Signature > >
- struct std::is_bind_expression< const volatile _Bind_result< _Result, _Signature > >
- struct std::is_bind_expression< volatile _Bind< _Signature > >
- struct std::is_bind_expression< volatile _Bind_result< _Result, _Signature > >
- struct std::is_placeholder< _Tp >
- struct std::is_placeholder< _Placeholder< _Num > >

Namespaces

- namespace std
- namespace std::placeholders

Macros

- #define **__glibcxx_want_bind_back**
- #define **__glibcxx_want_bind_front**
- #define **__glibcxx_want_boyer_moore_searcher**
- #define **__glibcxx_want_constexpr_functional**
- #define **__glibcxx_want_copyable_function**
- #define **__glibcxx_want_function_ref**
- #define **__glibcxx_want_invoke**
- #define **__glibcxx_want_invoke_r**
- #define **__glibcxx_want_move_only_function**
- #define **__glibcxx_want_not_fn**
- #define **__glibcxx_want_ranges**
- #define **__glibcxx_want_reference_wrapper**
- #define **__glibcxx_want_transparent_operators**
- #define **_GLIBCXX_FUNCTIONAL**
- #define **_GLIBCXX_NOT_FN_CALL_OP**(_QUALS)
- #define **_GLIBCXX_PLACEHOLDER**

Typedefs

- template<typename _Tp, typename _Tp2 = typename decay<_Tp>::type>
  using **std::__is_socketlike**

Functions

- template<typename _Func, typename... _BoundArgs>
  constexpr _Bind_helper< __is_socketlike< _Func >::value, _Func, _BoundArgs... >::type std::bind (_Func &&↩
  __f, _BoundArgs &&... __args)
- template<typename _Result, typename _Func, typename... _BoundArgs>
  constexpr _Bindres_helper< _Result, _Func, _BoundArgs... >::type std::bind (_Func &&__f, _BoundArgs &&...
  __args)
- template<typename _Tp, typename _Class>
  constexpr _Mem_fn< _Tp _Class::* > std::mem_fn (_Tp _Class::*__pm) noexcept

Variables

- const _Placeholder< 1 > **std::placeholders::_1**
- const _Placeholder< 10 > **std::placeholders::_10**
- const _Placeholder< 11 > **std::placeholders::_11**
- const _Placeholder< 12 > **std::placeholders::_12**
- const _Placeholder< 13 > **std::placeholders::_13**
- const _Placeholder< 14 > **std::placeholders::_14**
- const _Placeholder< 15 > **std::placeholders::_15**
- const _Placeholder< 16 > **std::placeholders::_16**
- const _Placeholder< 17 > **std::placeholders::_17**
- const _Placeholder< 18 > **std::placeholders::_18**
- const _Placeholder< 19 > **std::placeholders::_19**
- const _Placeholder< 2 > **std::placeholders::_2**
- const _Placeholder< 20 > **std::placeholders::_20**
- const _Placeholder< 21 > **std::placeholders::_21**
- const _Placeholder< 22 > **std::placeholders::_22**
- const _Placeholder< 23 > **std::placeholders::_23**
- const _Placeholder< 24 > **std::placeholders::_24**
- const _Placeholder< 25 > **std::placeholders::_25**
- const _Placeholder< 26 > **std::placeholders::_26**
- const _Placeholder< 27 > **std::placeholders::_27**
- const _Placeholder< 28 > **std::placeholders::_28**
- const _Placeholder< 29 > **std::placeholders::_29**
- const _Placeholder< 3 > **std::placeholders::_3**
- const _Placeholder< 4 > **std::placeholders::_4**
- const _Placeholder< 5 > **std::placeholders::_5**
- const _Placeholder< 6 > **std::placeholders::_6**
- const _Placeholder< 7 > **std::placeholders::_7**
- const _Placeholder< 8 > **std::placeholders::_8**
- const _Placeholder< 9 > **std::placeholders::_9**
- template<typename _Tp>
  constexpr bool **std::is_bind_expression_v**
- template<typename _Tp>
  constexpr int **std::is_placeholder_v**

6.566.1 Detailed Description

This is a Standard C++ Library header.

6.567 future File Reference

Classes

- class std::__basic_future< _Res >
- class std::future< _Res >
- class std::future< _Res & >
- class std::future< void >
- class std::future_error
- struct std::is_error_code_enum< future_errc >
- class std::packaged_task< _Res(_ArgTypes...)>
- class std::promise< _Res >
- class std::promise< _Res & >
- class std::promise< void >
- class std::shared_future< _Res >
- class std::shared_future< _Res & >
- class std::shared_future< void >

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_FUTURE**

Enumerations

- enum class std::future_errc { **future_already_retrieved** , **promise_already_satisfied** , **no_state** , **broken_promise** }
- enum class std::future_status { **ready** , **timeout** , **deferred** }
- enum class std::launch { **async** , **deferred** }

Functions

- template<typename _Fn, typename... _Args>
  future< __async_result_of< _Fn, _Args... > > std::async (_Fn &&__fn, _Args &&... __args)
- template<typename _Fn, typename... _Args>
  future< __async_result_of< _Fn, _Args... > > std::async (launch __policy, _Fn &&__fn, _Args &&... __args)
- const error_category & std::future_category () noexcept
- error_code std::make_error_code (future_errc __errc) noexcept
- error_condition std::make_error_condition (future_errc __errc) noexcept
- constexpr launch **std::operator&** (launch __x, launch __y) noexcept
- constexpr launch & **std::operator&=** (launch &__x, launch __y) noexcept
- constexpr launch **std::operator^** (launch __x, launch __y) noexcept
- constexpr launch & **std::operator^=** (launch &__x, launch __y) noexcept
- constexpr launch **std::operator|** (launch __x, launch __y) noexcept
- constexpr launch & **std::operator|=** (launch &__x, launch __y) noexcept
- constexpr launch **std::operator~** (launch __x) noexcept
- template<typename _Fun, typename _Signature = __function_guide_t<_Fun, decltype(&_Fun::operator())>>
  std::packaged_task (_Fun) -> packaged_task< _Signature >
- template<typename _Res, typename... _ArgTypes>
  std::packaged_task (_Res(*)(_ArgTypes...)) -> packaged_task< _Res(_ArgTypes...)>

- template<typename _Res, typename... _ArgTypes>
  void std::swap (packaged_task< _Res(_ArgTypes...)> &__x, packaged_task< _Res(_ArgTypes...)> &__y) noexcept
- template<typename _Res>
  void **std::swap** (promise< _Res > &__x, promise< _Res > &__y) noexcept

6.567.1 Detailed Description

This is a Standard C++ Library header.

6.568 generator File Reference

6.568.1 Detailed Description

This is a Standard C++ Library header.

6.569 inplace_vector File Reference

Macros

- #define **__glibcxx_want_inplace_vector**
- #define **_GLIBCXX_INPLACE_VECTOR**

6.569.1 Detailed Description

This is a Standard C++ Library header.

6.570 iomanip File Reference

Namespaces

- namespace std

Macros

- #define **__glibcxx_want_quoted_string_io**
- #define **_GLIBCXX_IOMANIP**

Functions

- template<typename _MoneyT>
  _Get_money< _MoneyT > std::get_money (_MoneyT &__mon, bool __intl=false)
- template<typename _CharT>
  _Get_time< _CharT > std::get_time (std::tm *__tmb, const _CharT *__fmt)
- template<typename _CharT, typename _Traits, typename _MoneyT>
  basic_ostream< _CharT, _Traits > & **std::operator**<< (basic_ostream< _CharT, _Traits > &__os, _Put_money< _MoneyT > __f)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::operator**<< (basic_ostream< _CharT, _Traits > &__os, _Put_time< _CharT > __f)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::operator**<< (basic_ostream< _CharT, _Traits > &__os, _Resetiosflags __f)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::operator**<< (basic_ostream< _CharT, _Traits > &__os, _Setbase __f)

- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::operator**<< (basic_ostream< _CharT, _Traits > &__os, _Setfill< _CharT > __f)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::operator**<< (basic_ostream< _CharT, _Traits > &__os, _Setiosflags __f)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::operator**<< (basic_ostream< _CharT, _Traits > &__os, _Setprecision __f)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::operator**<< (basic_ostream< _CharT, _Traits > &__os, _Setw __f)
- template ostream & **std::operator**<< (ostream &, _Resetiosflags)
- template ostream & **std::operator**<< (ostream &, _Setbase)
- template ostream & **std::operator**<< (ostream &, _Setfill< char >)
- template ostream & **std::operator**<< (ostream &, _Setiosflags)
- template ostream & **std::operator**<< (ostream &, _Setprecision)
- template ostream & **std::operator**<< (ostream &, _Setw)
- template wostream & **std::operator**<< (wostream &, _Resetiosflags)
- template wostream & **std::operator**<< (wostream &, _Setbase)
- template wostream & **std::operator**<< (wostream &, _Setfill< wchar_t >)
- template wostream & **std::operator**<< (wostream &, _Setiosflags)
- template wostream & **std::operator**<< (wostream &, _Setprecision)
- template wostream & **std::operator**<< (wostream &, _Setw)
- template<typename _CharT, typename _Traits, typename _MoneyT>
  basic_istream< _CharT, _Traits > & **std::operator**>> (basic_istream< _CharT, _Traits > &__is, _Get_money< _MoneyT > __f)
- template<typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & **std::operator**>> (basic_istream< _CharT, _Traits > &__is, _Get_time< _CharT > __f)
- template<typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & **std::operator**>> (basic_istream< _CharT, _Traits > &__is, _Resetiosflags __f)
- template<typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & **std::operator**>> (basic_istream< _CharT, _Traits > &__is, _Setbase __f)
- template<typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & **std::operator**>> (basic_istream< _CharT, _Traits > &__is, _Setfill< _CharT > __f)
- template<typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & **std::operator**>> (basic_istream< _CharT, _Traits > &__is, _Setiosflags __f)
- template<typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & **std::operator**>> (basic_istream< _CharT, _Traits > &__is, _Setprecision __f)
- template<typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & **std::operator**>> (basic_istream< _CharT, _Traits > &__is, _Setw __f)
- template istream & **std::operator**>> (istream &, _Resetiosflags)
- template istream & **std::operator**>> (istream &, _Setbase)
- template istream & **std::operator**>> (istream &, _Setfill< char >)
- template istream & **std::operator**>> (istream &, _Setiosflags)
- template istream & **std::operator**>> (istream &, _Setprecision)
- template istream & **std::operator**>> (istream &, _Setw)
- template wistream & **std::operator**>> (wistream &, _Resetiosflags)

- template wistream & **std::operator>>** (wistream &, _Setbase)
- template wistream & **std::operator>>** (wistream &, _Setfill< wchar_t >)
- template wistream & **std::operator>>** (wistream &, _Setiosflags)
- template wistream & **std::operator>>** (wistream &, _Setprecision)
- template wistream & **std::operator>>** (wistream &, _Setw)
- template<typename _MoneyT>
  _Put_money< _MoneyT > std::put_money (const _MoneyT &__mon, bool __intl=false)
- template<typename _CharT>
  _Put_time< _CharT > std::put_time (const std::tm *__tmb, const _CharT *__fmt)
- _Resetiosflags std::resetiosflags (ios_base::fmtflags __mask)
- _Setbase std::setbase (int __base)
- template<typename _CharT>
  _Setfill< _CharT > std::setfill (_CharT __c)
- _Setiosflags std::setiosflags (ios_base::fmtflags __mask)
- _Setprecision std::setprecision (int __n)
- _Setw std::setw (int __n)

6.570.1 Detailed Description

This is a Standard C++ Library header.

6.571 ios File Reference

Macros

- #define **__glibcxx_want_ios_noreplace**
- #define **_GLIBCXX_IOS**

6.571.1 Detailed Description

This is a Standard C++ Library header.

6.572 iosfwd File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_IOSFWD**

Typedefs

- typedef basic_filebuf< char > std::filebuf
- typedef basic_fstream< char > std::fstream
- typedef basic_ifstream< char > std::ifstream
- typedef basic_ios< char > std::ios
- typedef basic_iostream< char > std::iostream
- typedef basic_istream< char > std::istream
- typedef basic_istringstream< char > std::istringstream
- typedef basic_ofstream< char > std::ofstream
- typedef basic_ostream< char > std::ostream
- typedef basic_ostringstream< char > std::ostringstream

- typedef basic_streambuf< char > std::streambuf
- typedef basic_stringbuf< char > std::stringbuf
- typedef basic_stringstream< char > std::stringstream
- typedef basic_filebuf< wchar_t > std::wfilebuf
- typedef basic_fstream< wchar_t > std::wfstream
- typedef basic_ifstream< wchar_t > std::wifstream
- typedef basic_ios< wchar_t > std::wios
- typedef basic_iostream< wchar_t > std::wiostream
- typedef basic_istream< wchar_t > std::wistream
- typedef basic_istringstream< wchar_t > std::wistringstream
- typedef basic_ofstream< wchar_t > std::wofstream
- typedef basic_ostream< wchar_t > std::wostream
- typedef basic_ostringstream< wchar_t > std::wostringstream
- typedef basic_streambuf< wchar_t > std::wstreambuf
- typedef basic_stringbuf< wchar_t > std::wstringbuf
- typedef basic_stringstream< wchar_t > std::wstringstream

6.572.1 Detailed Description

This is a Standard C++ Library header.

6.573 iostream File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_IOSTREAM**

Variables

- static ios_base::Init **std::__ioinit**

Standard Stream Objects

The `<iostream>` *header declares the eight standard stream objects. For other declarations, see* `https://gcc.gnu.org/onlinedocs/libstdc++/manual/io.html` *and the I/O forward declarations*

They are required by default to cooperate with the global C library's `FILE` *streams, and to be available during program startup and termination. For more information, see the section of the manual linked to above.*

- ostream std::cerr
- istream std::cin
- ostream std::clog
- ostream std::cout
- wostream std::wcerr
- wistream std::wcin
- wostream std::wclog
- wostream std::wcout

6.573.1 Detailed Description

This is a Standard C++ Library header.

6.574 istream File Reference

Classes

- class std::basic_iostream< _CharT, _Traits >
- class std::basic_istream< _CharT, _Traits >
- class std::basic_istream< _CharT, _Traits >::sentry

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_ISTREAM**

Typedefs

- template<typename _Is, typename _Tp>
  using **std::__rvalue_stream_extraction_t**

Functions

- template<typename _CharT, typename _Traits>
  void **std::__istream_extract** (basic_istream< _CharT, _Traits > &, _CharT *, streamsize)
- void **std::__istream_extract** (istream &, char *, streamsize)
- template<typename _Istream, typename _Tp>
  __rvalue_stream_extraction_t< _Istream, _Tp > std::operator>> (_Istream &&__is, _Tp &&__x)
- template<typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & std::ws (basic_istream< _CharT, _Traits > &__is)
- template<typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__in, _CharT &__c)
- template<class _Traits>
  basic_istream< char, _Traits > & std::operator>> (basic_istream< char, _Traits > &__in, signed char &__c)
- template<class _Traits>
  basic_istream< char, _Traits > & std::operator>> (basic_istream< char, _Traits > &__in, unsigned char &__c)
- template<typename _CharT, typename _Traits, size_t _Num>
  basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__in, _CharT(&__s)[_Num])
- template<class _Traits, size_t _Num>
  basic_istream< char, _Traits > & std::operator>> (basic_istream< char, _Traits > &__in, signed char(&__s)[_Num])
- template<class _Traits, size_t _Num>
  basic_istream< char, _Traits > & std::operator>> (basic_istream< char, _Traits > &__in, unsigned char(&__s)[_Num])

6.574.1 Detailed Description

This is a Standard C++ Library header.

6.575 experimental/iterator File Reference

Classes

- class std::experimental::fundamentals_v2::ostream_joiner< _DelimT, _CharT, _Traits >

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **__cpp_lib_experimental_ostream_joiner**
- #define **_GLIBCXX_EXPERIMENTAL_ITERATOR**

Functions

- template<typename _CharT, typename _Traits, typename _DelimT>
  ostream_joiner< decay_t< _DelimT >, _CharT, _Traits > std::experimental::make_ostream_joiner (basic_ostream< _CharT, _Traits > &__os, _DelimT &&__delimiter)

6.575.1 Detailed Description

This is a TS C++ Library header.

6.576 ext/iterator File Reference

Namespaces

- namespace __gnu_cxx

Macros

- #define **_EXT_ITERATOR**

Functions

- template<typename _InputIterator, typename _Distance>
  void **__gnu_cxx::__distance** (_InputIterator __first, _InputIterator __last, _Distance &__n, std::input_iterator_tag)
- template<typename _RandomAccessIterator, typename _Distance>
  void **__gnu_cxx::__distance** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Distance &__n, std::random_access_iterator_tag)
- template<typename _InputIterator, typename _Distance>
  void __gnu_cxx::distance (_InputIterator __first, _InputIterator __last, _Distance &__n)

6.576.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

6.577 iterator File Reference

Macros

- #define **__glibcxx_want_array_constexpr**
- #define **__glibcxx_want_constexpr_iterator**

- #define **__glibcxx_want_make_reverse_iterator**
- #define **__glibcxx_want_move_iterator_concept**
- #define **__glibcxx_want_nonmember_container_access**
- #define **__glibcxx_want_null_iterators**
- #define **__glibcxx_want_ranges**
- #define **__glibcxx_want_ssize**
- #define **_GLIBCXX_ITERATOR**

6.577.1 Detailed Description

This is a Standard C++ Library header.

6.578 latch File Reference

Macros

- #define **__glibcxx_want_latch**
- #define **_GLIBCXX_LATCH**

6.578.1 Detailed Description

This is a Standard C++ Library header.

6.579 limits File Reference

Classes

- struct std::__numeric_limits_base
- struct std::numeric_limits< _Tp >
- struct std::numeric_limits< bool >
- struct std::numeric_limits< char >
- struct std::numeric_limits< char16_t >
- struct std::numeric_limits< char32_t >
- struct std::numeric_limits< double >
- struct std::numeric_limits< float >
- struct std::numeric_limits< int >
- struct std::numeric_limits< long >
- struct std::numeric_limits< long double >
- struct std::numeric_limits< long long >
- struct std::numeric_limits< short >
- struct std::numeric_limits< signed char >
- struct std::numeric_limits< unsigned char >
- struct std::numeric_limits< unsigned int >
- struct std::numeric_limits< unsigned long >
- struct std::numeric_limits< unsigned long long >
- struct std::numeric_limits< unsigned short >
- struct std::numeric_limits< wchar_t >

Namespaces

- namespace std

Macros

- #define **__glibcxx_concat3**(P, M, S)
- #define **__glibcxx_concat3_**(P, M, S)
- #define **__glibcxx_digits**(T)
- #define **__glibcxx_digits10**(T)
- #define **__glibcxx_digits10_b**(T, B)
- #define **__glibcxx_digits_b**(T, B)
- #define **__glibcxx_double_has_denorm_loss**
- #define **__glibcxx_double_tinyness_before**
- #define **__glibcxx_double_traps**
- #define **__glibcxx_float_has_denorm_loss**
- #define **__glibcxx_float_n**(BITSIZE)
- #define **__glibcxx_float_tinyness_before**
- #define **__glibcxx_float_traps**
- #define **__glibcxx_integral_traps**
- #define **__glibcxx_long_double_has_denorm_loss**
- #define **__glibcxx_long_double_tinyness_before**
- #define **__glibcxx_long_double_traps**
- #define **__glibcxx_max**(T)
- #define **__glibcxx_max_b**(T, B)
- #define **__glibcxx_max_digits10**(T)
- #define **__glibcxx_min**(T)
- #define **__glibcxx_min_b**(T, B)
- #define **__glibcxx_signed**(T)
- #define **__glibcxx_signed_b**(T, B)
- #define **__INT_N**(TYPE, BITSIZE, EXT, UEXT)
- #define **__INT_N_201103**(TYPE)
- #define **__INT_N_U201103**(TYPE)
- #define **__max_digits10**
- #define **_GLIBCXX_NUMERIC_LIMITS**

Enumerations

- enum std::float_denorm_style { std::denorm_indeterminate , std::denorm_absent , std::denorm_present }
- enum std::float_round_style { std::round_indeterminate , std::round_toward_zero , std::round_to_nearest , std::round_toward_infinity , std::round_toward_neg_infinity }

6.579.1 Detailed Description

This is a Standard C++ Library header.

6.580 debug/list File Reference

Classes

- class std::__debug::list< _Tp, _Allocator >

Namespaces

- namespace __gnu_debug
- namespace std
- namespace std::__debug

Macros

- #define **_GLIBCXX20_ONLY**(__expr)
- #define **_GLIBCXX_DEBUG_LIST**
- #define **_GLIBCXX_LIST_REMOVE_RETURN_TYPE_TAG**

Functions

- template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  std::__debug::list (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> list< _ValT, _Allocator >
- template<typename _Tp, typename _Allocator = allocator<_Tp>, typename = _RequireAllocator<_Allocator>>
  std::__debug::list (size_t, _Tp, _Allocator=_Allocator()) -> list< _Tp, _Allocator >
- template<typename _Tp, typename _Alloc>
  constexpr __detail::__synth3way_t< _Tp > **std::__debug::operator<=>** (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)
- template<typename _Tp, typename _Alloc>
  bool **std::__debug::operator==** (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)
- template<typename _Tp, typename _Alloc>
  void **std::__debug::swap** (list< _Tp, _Alloc > &__lhs, list< _Tp, _Alloc > &__rhs) noexcept(/*conditional */)

6.580.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

6.581 experimental/list File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **_GLIBCXX_EXPERIMENTAL_LIST**

Typedefs

- template<typename _Tp>
  using **std::experimental::fundamentals_v2::pmr::list**

Functions

- template<typename _Tp, typename _Alloc, typename _Up>
  void **std::experimental::erase** (list< _Tp, _Alloc > &__cont, const _Up &__value)
- template<typename _Tp, typename _Alloc, typename _Predicate>
  void **std::experimental::erase_if** (list< _Tp, _Alloc > &__cont, _Predicate __pred)

6.581.1 Detailed Description

This is a TS C++ Library header.

6.582 list File Reference

Namespaces

- namespace std

Macros

- #define **__glibcxx_want_algorithm_default_value_type**
- #define **__glibcxx_want_allocator_traits_is_always_equal**
- #define **__glibcxx_want_containers_ranges**
- #define **__glibcxx_want_erase_if**
- #define **__glibcxx_want_incomplete_container_elements**
- #define **__glibcxx_want_list_remove_return_type**
- #define **__glibcxx_want_nonmember_container_access**
- #define **_GLIBCXX_LIST**

Typedefs

- template<typename _Tp>
  using **std::pmr::list**

6.582.1 Detailed Description

This is a Standard C++ Library header.

6.583 locale File Reference

Macros

- #define **_GLIBCXX_LOCALE**

6.583.1 Detailed Description

This is a Standard C++ Library header.

6.584 debug/map File Reference

Namespaces

- namespace std
- namespace std::__debug

Macros

- #define **_GLIBCXX_DEBUG_MAP**

6.584.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

6.585 experimental/map File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **_GLIBCXX_EXPERIMENTAL_MAP**

Typedefs

- template<typename _Key, typename _Tp, typename _Compare = less<_Key>>
  using **std::experimental::fundamentals_v2::pmr::map**
- template<typename _Key, typename _Tp, typename _Compare = less<_Key>>
  using **std::experimental::fundamentals_v2::pmr::multimap**

Functions

- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc, typename _Predicate>
  void **std::experimental::erase_if** (map< _Key, _Tp, _Compare, _Alloc > &__cont, _Predicate __pred)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc, typename _Predicate>
  void **std::experimental::erase_if** (multimap< _Key, _Tp, _Compare, _Alloc > &__cont, _Predicate __pred)

6.585.1 Detailed Description

This is a TS C++ Library header.

6.586 map File Reference

Namespaces

- namespace std

Macros

- #define **__glibcxx_want_allocator_traits_is_always_equal**
- #define **__glibcxx_want_containers_ranges**
- #define **__glibcxx_want_erase_if**
- #define **__glibcxx_want_generic_associative_lookup**
- #define **__glibcxx_want_map_try_emplace**
- #define **__glibcxx_want_node_extract**
- #define **__glibcxx_want_nonmember_container_access**
- #define **__glibcxx_want_tuple_like**
- #define **_GLIBCXX_MAP**

Typedefs

- template<typename _Key, typename _Tp, typename _Cmp = std::less<_Key>>
  using **std::pmr::map**
- template<typename _Key, typename _Tp, typename _Cmp = std::less<_Key>>
  using **std::pmr::multimap**

Functions

- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc, typename _Predicate>
  map< _Key, _Tp, _Compare, _Alloc >::size_type **std::erase_if** (map< _Key, _Tp, _Compare, _Alloc > &__cont, _Predicate __pred)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc, typename _Predicate>
  multimap< _Key, _Tp, _Compare, _Alloc >::size_type **std::erase_if** (multimap< _Key, _Tp, _Compare, _Alloc > &__cont, _Predicate __pred)

6.586.1 Detailed Description

This is a Standard C++ Library header.

6.587 math.h File Reference

Functions

- template<typename _Tp>
  _Tp abs (const complex< _Tp > &)
- template<typename _Tp>
  std::complex< _Tp > **acos** (const std::complex< _Tp > &)
- template<typename _Tp>
  std::complex< _Tp > **acosh** (const std::complex< _Tp > &)
- template<typename _Tp>
  std::complex< _Tp > **asin** (const std::complex< _Tp > &)
- template<typename _Tp>
  std::complex< _Tp > **asinh** (const std::complex< _Tp > &)
- template<typename _Tp>
  std::complex< _Tp > **atan** (const std::complex< _Tp > &)
- constexpr float **atan2** (float __y, float __x)
- template<typename _Tp>
  std::complex< _Tp > **atanh** (const std::complex< _Tp > &)
- constexpr float **cbrt** (float __x)
- constexpr float **ceil** (float __x)
- constexpr float **copysign** (float __x, float __y)
- template<typename _Tp>
  complex< _Tp > cos (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > cosh (const complex< _Tp > &)
- constexpr float **erf** (float __x)
- constexpr float **erfc** (float __x)
- template<typename _Tp>
  complex< _Tp > exp (const complex< _Tp > &)
- constexpr float **exp2** (float __x)
- constexpr float **expm1** (float __x)
- template<typename _Tp>
  _Tp fabs (const std::complex< _Tp > &__z)
- constexpr float **fdim** (float __x, float __y)
- constexpr float **floor** (float __x)
- constexpr float **fma** (float __x, float __y, float __z)
- constexpr float **fmax** (float __x, float __y)
- constexpr float **fmin** (float __x, float __y)
- constexpr float **fmod** (float __x, float __y)
- constexpr int **fpclassify** (float __x)
- float **frexp** (float __x, int *__exp)
- constexpr float **hypot** (float __x, float __y)
- constexpr int **ilogb** (float __x)
- constexpr bool **isfinite** (float __x)
- constexpr bool **isgreater** (float __x, float __y)
- constexpr bool **isgreaterequal** (float __x, float __y)
- constexpr bool **isinf** (float __x)
- constexpr bool **isless** (float __x, float __y)
- constexpr bool **islessequal** (float __x, float __y)
- constexpr bool **islessgreater** (float __x, float __y)
- constexpr bool **isnan** (float __x)

- constexpr bool **isnormal** (float __x)
- constexpr bool **isunordered** (float __x, float __y)
- constexpr float **ldexp** (float __x, int __exp)
- constexpr float **lgamma** (float __x)
- constexpr long long **llrint** (float __x)
- constexpr long long **llround** (float __x)
- template<typename _Tp>
  complex< _Tp > log (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > log10 (const complex< _Tp > &)
- constexpr float **log1p** (float __x)
- constexpr float **log2** (float __x)
- constexpr float **logb** (float __x)
- constexpr long **lrint** (float __x)
- constexpr long **lround** (float __x)
- float **modf** (float __x, float *__iptr)
- constexpr float **nearbyint** (float __x)
- constexpr float **nextafter** (float __x, float __y)
- constexpr float **nexttoward** (float __x, long double __y)
- template<typename _Tp>
  complex< _Tp > pow (const complex< _Tp > &, int)
- constexpr float **remainder** (float __x, float __y)
- float **remquo** (float __x, float __y, int *__pquo)
- constexpr float **rint** (float __x)
- constexpr float **round** (float __x)
- constexpr float **scalbln** (float __x, long __ex)
- constexpr float **scalbn** (float __x, int __ex)
- constexpr bool **signbit** (float __x)
- template<typename _Tp>
  complex< _Tp > sin (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > sinh (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > sqrt (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > tan (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > tanh (const complex< _Tp > &)
- constexpr float **tgamma** (float __x)
- constexpr float **trunc** (float __x)

6.587.1 Detailed Description

This is a Standard C++ Library header.

6.587.2 Function Documentation

abs()

```
template<typename _Tp>
_Tp std::abs (
            const complex< _Tp > & __z)  [inline]
```

Return magnitude of *z*.

cos()

```
template<typename _Tp>
complex< _Tp > std::cos (
            const complex< _Tp > & __z)  [inline]
```

Return complex cosine of *z*.

cosh()

```
template<typename _Tp>
complex< _Tp > std::cosh (
            const complex< _Tp > & __z)  [inline]
```

Return complex hyperbolic cosine of *z*.

exp()

```
template<typename _Tp>
complex< _Tp > std::exp (
            const complex< _Tp > & __z)  [inline]
```

Return complex base e exponential of *z*.

fabs()

```
template<typename _Tp>
_Tp std::fabs (
            const std::complex< _Tp > & __z)  [inline]
```

fabs(__z) TR1 8.1.8 [tr.c99.cmplx.fabs]

log()

```
template<typename _Tp>
complex< _Tp > std::log (
            const complex< _Tp > & __z)  [inline]
```

Return complex natural logarithm of *z*.

log10()

```
template<typename _Tp>
complex< _Tp > std::log10 (
            const complex< _Tp > & __z)  [inline]
```

Return complex base 10 logarithm of *z*.

pow()

```
template<typename _Tp>
complex< _Tp > std::pow (
            const complex< _Tp > & __z,
            int __n)  [inline]
```

Return *x* to the *y'th* power.

sin()

```
template<typename _Tp>
complex< _Tp > std::sin (
            const complex< _Tp > & __z)  [inline]
```

Return complex sine of *z*.

sinh()

```
template<typename _Tp>
complex< _Tp > std::sinh (
            const complex< _Tp > & __z)  [inline]
```

Return complex hyperbolic sine of *z*.

sqrt()

```
template<typename _Tp>
complex< _Tp > std::sqrt (
            const complex< _Tp > & __z)  [inline]
```

Return complex square root of *z*.

tan()

```
template<typename _Tp>
complex< _Tp > std::tan (
            const complex< _Tp > & __z)  [inline]
```

Return complex tangent of *z*.

tanh()

```
template<typename _Tp>
complex< _Tp > std::tanh (
            const complex< _Tp > & __z)  [inline]
```

Return complex hyperbolic tangent of *z*.

6.588 mdspan File Reference

Macros

- #define **__glibcxx_want_aligned_accessor**
- #define **__glibcxx_want_mdspan**
- #define **_GLIBCXX_MDSPAN**

6.588.1 Detailed Description

This is a Standard C++ Library header.

6.589 experimental/memory File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **__cpp_lib_experimental_observer_ptr**
- #define **_GLIBCXX_EXPERIMENTAL_MEMORY**

Functions

- template<typename _Tp>
  constexpr observer_ptr< _Tp > **std::experimental::make_observer** (_Tp *__p) noexcept
- template<typename _Tp>
  constexpr bool **std::experimental::operator!=** (nullptr_t, observer_ptr< _Tp > __p) noexcept
- template<typename _Tp>
  bool **std::experimental::operator!=** (observer_ptr< _Tp > __p, nullptr_t) noexcept
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator!=** (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator**< (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator**<= (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)
- template<typename _Tp>
  constexpr bool **std::experimental::operator==** (nullptr_t, observer_ptr< _Tp > __p) noexcept
- template<typename _Tp>
  constexpr bool **std::experimental::operator==** (observer_ptr< _Tp > __p, nullptr_t) noexcept
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator==** (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator**> (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator**>= (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)
- template<typename _Tp>
  constexpr void **std::experimental::swap** (observer_ptr< _Tp > &__p1, observer_ptr< _Tp > &__p2) noexcept

6.589.1 Detailed Description

This is a TS C++ Library header.

6.590 ext/memory File Reference

Classes

- class __gnu_cxx::_Temporary_buffer< _ForwardIterator, _Tp >
- struct __gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >

Namespaces

- namespace __gnu_cxx

Macros

- #define **_EXT_MEMORY**

Functions

- template<typename _InputIter, typename _Size, typename _ForwardIter>
  std::pair< _InputIter, _ForwardIter > **__gnu_cxx::__uninitialized_copy_n** (_InputIter __first, _Size __count, ↩
  _ForwardIter __result)
- template<typename _InputIter, typename _Size, typename _ForwardIter>
  std::pair< _InputIter, _ForwardIter > **__gnu_cxx::__uninitialized_copy_n** (_InputIter __first, _Size __count, ↩
  _ForwardIter __result, std::input_iterator_tag)

- template<typename _RandomAccessIter, typename _Size, typename _ForwardIter>
  std::pair< _RandomAccessIter, _ForwardIter > **__gnu_cxx::__uninitialized_copy_n** (_RandomAccessIter _↩
  _first, _Size __count, _ForwardIter __result, std::random_access_iterator_tag)
- template<typename _InputIter, typename _Size, typename _ForwardIter, typename _Allocator>
  std::pair< _InputIter, _ForwardIter > **__gnu_cxx::__uninitialized_copy_n_a** (_InputIter __first, _Size __count, _ForwardIter __result, _Allocator __alloc)
- template<typename _InputIter, typename _Size, typename _ForwardIter, typename _Tp>
  std::pair< _InputIter, _ForwardIter > **__gnu_cxx::__uninitialized_copy_n_a** (_InputIter __first, _Size __count, _ForwardIter __result, std::allocator< _Tp >)
- template<typename _InputIter, typename _Size, typename _ForwardIter>
  std::pair< _InputIter, _ForwardIter > __gnu_cxx::uninitialized_copy_n (_InputIter __first, _Size __count, _↩
  ForwardIter __result)

6.590.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

6.591 memory File Reference

Namespaces

- namespace std

Macros

- #define **__glibcxx_want_addressof_constexpr**
- #define **__glibcxx_want_allocator_traits_is_always_equal**
- #define **__glibcxx_want_assume_aligned**
- #define **__glibcxx_want_atomic_shared_ptr**
- #define **__glibcxx_want_atomic_value_initialization**
- #define **__glibcxx_want_constexpr_dynamic_alloc**
- #define **__glibcxx_want_constexpr_memory**
- #define **__glibcxx_want_enable_shared_from_this**
- #define **__glibcxx_want_indirect**
- #define **__glibcxx_want_is_sufficiently_aligned**
- #define **__glibcxx_want_make_unique**
- #define **__glibcxx_want_out_ptr**
- #define **__glibcxx_want_parallel_algorithm**
- #define **__glibcxx_want_polymorphic**
- #define **__glibcxx_want_ranges**
- #define **__glibcxx_want_raw_memory_algorithms**
- #define **__glibcxx_want_shared_ptr_arrays**
- #define **__glibcxx_want_shared_ptr_weak_type**
- #define **__glibcxx_want_smart_ptr_for_overwrite**
- #define **__glibcxx_want_smart_ptr_owner_equality**
- #define **__glibcxx_want_start_lifetime_as**
- #define **__glibcxx_want_to_address**
- #define **__glibcxx_want_transparent_operators**
- #define **_GLIBCXX_MEMORY**

Enumerations

- enum class std::pointer_safety { **relaxed** , **preferred** , **strict** }

Functions

- void std::declare_no_pointers (char *, size_t)
- void std::declare_reachable (void *)
- pointer_safety std::get_pointer_safety () noexcept
- void std::undeclare_no_pointers (char *, size_t)
- template<typename _Tp>
  _Tp * std::undeclare_reachable (_Tp *__p)

6.591.1 Detailed Description

This is a Standard C++ Library header.

6.592 experimental/memory_resource File Reference

Namespaces

- namespace __gnu_cxx
- namespace std
- namespace std::experimental

Macros

- #define **__cpp_lib_experimental_memory_resources**
- #define **_GLIBCXX_EXPERIMENTAL_MEMORY_RESOURCE**
- #define **_GLIBCXX_MAX_ALIGN_MATCHES_MALLOC**

Typedefs

- template<typename _Alloc>
  using **std::experimental::fundamentals_v2::pmr::resource_adaptor**

Functions

- memory_resource * std::experimental::fundamentals_v2::pmr::get_default_resource () noexcept
- memory_resource * **std::experimental::fundamentals_v2::pmr::new_delete_resource** () noexcept
- memory_resource * **std::experimental::fundamentals_v2::pmr::null_memory_resource** () noexcept
- bool **std::experimental::fundamentals_v2::pmr::operator!=** (const memory_resource &__a, const memory_resource &__b) noexcept
- template<class _Tp1, class _Tp2>
  bool **std::experimental::fundamentals_v2::pmr::operator!=** (const polymorphic_allocator< _Tp1 > &__a, const polymorphic_allocator< _Tp2 > &__b) noexcept
- bool **std::experimental::fundamentals_v2::pmr::operator==** (const memory_resource &__a, const memory_resource &__b) noexcept
- template<class _Tp1, class _Tp2>
  bool **std::experimental::fundamentals_v2::pmr::operator==** (const polymorphic_allocator< _Tp1 > &__a, const polymorphic_allocator< _Tp2 > &__b) noexcept
- memory_resource * std::experimental::fundamentals_v2::pmr::set_default_resource (memory_resource *__r) noexcept

6.592.1 Detailed Description

This is a TS C++ Library header.

6.592.2 Function Documentation

get_default_resource()

```
memory_resource * std::experimental::fundamentals_v2::pmr::get_default_resource ()  [inline],
[noexcept]
```

Get the current default resource.

set_default_resource()

```
memory_resource * std::experimental::fundamentals_v2::pmr::set_default_resource (
            memory_resource * __r)  [inline], [noexcept]
```

Change the default resource and return the previous one.

6.593 memory_resource File Reference

Classes

- class std::pmr::monotonic_buffer_resource
- struct std::pmr::pool_options
- class std::pmr::unsynchronized_pool_resource

Namespaces

- namespace std

Macros

- #define **__glibcxx_want_memory_resource**
- #define **__glibcxx_want_polymorphic_allocator**
- #define **_GLIBCXX_MEMORY_RESOURCE**

Functions

- memory_resource * std::pmr::get_default_resource () noexcept
- memory_resource * std::pmr::new_delete_resource () noexcept
- memory_resource * std::pmr::null_memory_resource () noexcept
- memory_resource * std::pmr::set_default_resource (memory_resource *__r) noexcept

6.593.1 Detailed Description

This is a Standard C++ Library header.
This header declares the pmr (std::pmr) memory resources.

6.593.2 Function Documentation

get_default_resource()

```
memory_resource * std::pmr::get_default_resource ()  [noexcept]
```

Get the current default memory resource pointer.

null_memory_resource()

```
memory_resource * std::pmr::null_memory_resource ()  [nodiscard], [noexcept]
```

A pmr::memory_resource that always throws `bad_alloc`

set_default_resource()

```
memory_resource * std::pmr::set_default_resource (
            memory_resource * __r)  [noexcept]
```

Replace the default memory resource pointer.

6.594 mutex File Reference

Classes

- struct std::once_flag
- class std::recursive_mutex
- class std::recursive_timed_mutex
- class std::timed_mutex

Namespaces

- namespace std

Macros

- #define **__glibcxx_want_scoped_lock**
- #define **_GLIBCXX_MUTEX**

Functions

- void **std::__once_proxy** (void)
- template<typename _Callable, typename... _Args>
  void std::call_once (once_flag &__once, _Callable &&__f, _Args &&... __args)
- template<typename _L1, typename _L2, typename... _L3>
  void std::lock (_L1 &__l1, _L2 &__l2, _L3 &... __l3)
- template<typename _L1, typename _L2, typename... _L3>
  int std::try_lock (_L1 &__l1, _L2 &__l2, _L3 &... __l3)

6.594.1 Detailed Description

This is a Standard C++ Library header.

6.595 numbers File Reference

Macros

- #define **__glibcxx_want_math_constants**
- #define **_GLIBCXX_NUMBERS**

6.595.1 Detailed Description

This is a Standard C++ Library header.

6.596 experimental/numeric File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **__cpp_lib_experimental_gcd_lcm**
- #define **_GLIBCXX_EXPERIMENTAL_NUMERIC**

Functions

- template<typename _Mn, typename _Nn>
  constexpr common_type_t< _Mn, _Nn > std::experimental::gcd (_Mn __m, _Nn __n) noexcept
- template<typename _Mn, typename _Nn>
  constexpr common_type_t< _Mn, _Nn > std::experimental::lcm (_Mn __m, _Nn __n)

6.596.1 Detailed Description

This is a TS C++ Library header.

6.597 ext/numeric File Reference

Namespaces

- namespace __gnu_cxx

Macros

- #define **_EXT_NUMERIC**

Functions

- template<typename _Tp, typename _Integer>
  _Tp **__gnu_cxx::__power** (_Tp __x, _Integer __n)
- template<typename _Tp, typename _Integer, typename _MonoidOperation>
  _Tp **__gnu_cxx::__power** (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)
- template<typename _ForwardIterator, typename _Tp>
  constexpr void __gnu_cxx::iota (_ForwardIterator __first, _ForwardIterator __last, _Tp __value)
- template<typename _Tp, typename _Integer>
  _Tp __gnu_cxx::power (_Tp __x, _Integer __n)
- template<typename _Tp, typename _Integer, typename _MonoidOperation>
  _Tp __gnu_cxx::power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)

6.597.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

6.598 numeric File Reference

Namespaces

- namespace std
- namespace std::__detail

Macros

- #define **__glibcxx_want_constexpr_numeric**
- #define **__glibcxx_want_gcd**
- #define **__glibcxx_want_gcd_lcm**

- #define **__glibcxx_want_interpolate**
- #define **__glibcxx_want_lcm**
- #define **__glibcxx_want_parallel_algorithm**
- #define **__glibcxx_want_ranges_iota**
- #define **__glibcxx_want_saturation_arithmetic**
- #define **_GLIBCXX_NUMERIC**
- #define **_PSTL_NUMERIC_FORWARD_DECLARED**

Functions

- template<typename _Res, typename _Tp>
  constexpr _Res **std::__detail::__abs_r** (_Tp __val)
- template<typename>
  void **std::__detail::__abs_r** (bool)=delete
- template<typename _Tp>
  constexpr _Tp **std::__detail::__gcd** (_Tp __m, _Tp __n)
- template<typename _InputIterator, typename _OutputIterator, typename _Tp>
  constexpr _OutputIterator std::exclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Tp __init)
- template<typename _InputIterator, typename _OutputIterator, typename _Tp, typename _BinaryOperation>
  constexpr _OutputIterator std::exclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Tp __init, _BinaryOperation __binary_op)
- template<typename _InputIterator, typename _OutputIterator>
  constexpr _OutputIterator std::inclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result)
- template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation>
  constexpr _OutputIterator std::inclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op)
- template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation, typename _Tp>
  constexpr _OutputIterator std::inclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op, _Tp __init)
- template<typename _InputIterator>
  constexpr iterator_traits< _InputIterator >::value_type std::reduce (_InputIterator __first, _InputIterator __last)
- template<typename _InputIterator, typename _Tp>
  constexpr _Tp std::reduce (_InputIterator __first, _InputIterator __last, _Tp __init)
- template<typename _InputIterator, typename _Tp, typename _BinaryOperation>
  constexpr _Tp std::reduce (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __binary_op)
- template<typename _InputIterator, typename _OutputIterator, typename _Tp, typename _BinaryOperation, typename _UnaryOperation>
  constexpr _OutputIterator std::transform_exclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Tp __init, _BinaryOperation __binary_op, _UnaryOperation __unary_op)
- template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation, typename _UnaryOperation>
  constexpr _OutputIterator std::transform_inclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op, _UnaryOperation __unary_op)
- template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation, typename _UnaryOperation, typename _Tp>
  constexpr _OutputIterator std::transform_inclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op, _UnaryOperation __unary_op, _Tp __init)
- template<typename _InputIterator, typename _Tp, typename _BinaryOperation, typename _UnaryOperation>
  constexpr _Tp std::transform_reduce (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __binary_op, _UnaryOperation __unary_op)
- template<typename _InputIterator1, typename _InputIterator2, typename _Tp>
  constexpr _Tp std::transform_reduce (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init)

- template<typename _InputIterator1, typename _InputIterator2, typename _Tp, typename _BinaryOperation1, typename _Binary↩ Operation2>
  constexpr _Tp std::transform_reduce (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init, _BinaryOperation1 __binary_op1, _BinaryOperation2 __binary_op2)

6.598.1 Detailed Description

This is a Standard C++ Library header.

6.599 parallel/numeric File Reference

Namespaces

- namespace std
- namespace std::__parallel

Macros

- #define **_GLIBCXX_PARALLEL_NUMERIC_H**

Functions

- template<typename __RAIter, typename _Tp, typename _BinaryOperation>
  _Tp **std::__parallel::__accumulate_switch** (__RAIter __begin, __RAIter __end, _Tp __init, _BinaryOperation __binary_op, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter, typename _Tp, typename _BinaryOperation, typename _IteratorTag>
  _Tp **std::__parallel::__accumulate_switch** (_IIter __begin, _IIter __end, _Tp __init, _BinaryOperation __↩ binary_op, _IteratorTag)
- template<typename _IIter, typename _Tp, typename _IteratorTag>
  _Tp **std::__parallel::__accumulate_switch** (_IIter __begin, _IIter __end, _Tp __init, _IteratorTag)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation, typename _IteratorTag1, typename _IteratorTag2>
  _OutputIterator **std::__parallel::__adjacent_difference_switch** (_IIter __begin, _IIter __end, _OutputIterator ↩ __result, _BinaryOperation __bin_op, _IteratorTag1, _IteratorTag2)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator **std::__parallel::__adjacent_difference_switch** (_IIter __begin, _IIter __end, _Output↩ Iterator __result, _BinaryOperation __bin_op, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter1, typename _IIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2, typename _↩ IteratorTag1, typename _IteratorTag2>
  _Tp **std::__parallel::__inner_product_switch** (_IIter1 __first1, _IIter1 __last1, _IIter2 __first2, _Tp __init, _↩ BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, _IteratorTag1, _IteratorTag2)
- template<typename _RAIter1, typename _RAIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>
  _Tp **std::__parallel::__inner_product_switch** (_RAIter1, _RAIter1, _RAIter2, _Tp, _BinaryFunction1, _Binary↩ Function2, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism=__gnu_parallel::parallel_unbal
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation, typename _IteratorTag1, typename _IteratorTag2>
  _OutputIterator **std::__parallel::__partial_sum_switch** (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __bin_op, _IteratorTag1, _IteratorTag2)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator **std::__parallel::__partial_sum_switch** (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __bin_op, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter, typename _Tp, typename _BinaryOperation>
  _Tp **std::__parallel::accumulate** (_IIter __begin, _IIter __end, _Tp __init, _BinaryOperation __binary_op)
- template<typename _IIter, typename _Tp, typename _BinaryOperation>
  _Tp **std::__parallel::accumulate** (_IIter __begin, _IIter __end, _Tp __init, _BinaryOperation __binary_op, __gnu_parallel::_Parallelism __parallelism_tag)

- template<typename _IIter, typename _Tp, typename _BinaryOperation>
  _Tp **std::__parallel::accumulate** (_IIter __begin, _IIter __end, _Tp __init, _BinaryOperation __binary_op, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _Tp>
  _Tp **std::__parallel::accumulate** (_IIter, _IIter, _Tp)
- template<typename _IIter, typename _Tp>
  _Tp **std::__parallel::accumulate** (_IIter, _IIter, _Tp, __gnu_parallel::_Parallelism)
- template<typename _IIter, typename _Tp>
  _Tp **std::__parallel::accumulate** (_IIter, _IIter, _Tp, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OutputIterator>
  _OutputIterator **std::__parallel::adjacent_difference** (_IIter __begin, _IIter __end, _OutputIterator __result)
- template<typename _IIter, typename _OutputIterator>
  _OutputIterator **std::__parallel::adjacent_difference** (_IIter __begin, _IIter __end, _OutputIterator __result, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter, typename _OutputIterator>
  _OutputIterator **std::__parallel::adjacent_difference** (_IIter __begin, _IIter __end, _OutputIterator __result, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator **std::__parallel::adjacent_difference** (_IIter __begin, _IIter __end, _OutputIterator __result, _↩
  BinaryOperation __bin_op, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator **std::__parallel::adjacent_difference** (_IIter __begin, _IIter __end, _OutputIterator __result, _↩
  BinaryOperation __binary_op)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator **std::__parallel::adjacent_difference** (_IIter __begin, _IIter __end, _OutputIterator __result, _↩
  BinaryOperation __binary_op, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter1, typename _IIter2, typename _Tp>
  _Tp **std::__parallel::inner_product** (_IIter1, _IIter1, _IIter2, _Tp)
- template<typename _IIter1, typename _IIter2, typename _Tp>
  _Tp **std::__parallel::inner_product** (_IIter1, _IIter1, _IIter2, _Tp, __gnu_parallel::_Parallelism)
- template<typename _IIter1, typename _IIter2, typename _Tp>
  _Tp **std::__parallel::inner_product** (_IIter1, _IIter1, _IIter2, _Tp, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>
  _Tp **std::__parallel::inner_product** (_IIter1, _IIter1, _IIter2, _Tp, _BinaryFunction1, _BinaryFunction2)
- template<typename _IIter1, typename _IIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>
  _Tp **std::__parallel::inner_product** (_IIter1, _IIter1, _IIter2, _Tp, _BinaryFunction1, _BinaryFunction2, __gnu_parallel::_Parallelism)
- template<typename _IIter1, typename _IIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>
  _Tp **std::__parallel::inner_product** (_IIter1, _IIter1, _IIter2, _Tp, _BinaryFunction1, _BinaryFunction2, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OutputIterator>
  _OutputIterator **std::__parallel::partial_sum** (_IIter __begin, _IIter __end, _OutputIterator __result)
- template<typename _IIter, typename _OutputIterator>
  _OutputIterator **std::__parallel::partial_sum** (_IIter __begin, _IIter __end, _OutputIterator __result, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator **std::__parallel::partial_sum** (_IIter __begin, _IIter __end, _OutputIterator __result, _Binary↩
  Operation __bin_op, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator **std::__parallel::partial_sum** (_IIter __begin, _IIter __end, _OutputIterator __result, _Binary↩
  Operation __binary_op)

6.599.1 Detailed Description

Parallel STL function calls corresponding to stl_numeric.h. The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

6.600 experimental/optional File Reference

Classes

- class std::experimental::fundamentals_v1::bad_optional_access
- struct std::hash< experimental::optional< _Tp > >
- struct std::experimental::fundamentals_v1::in_place_t
- struct std::experimental::fundamentals_v1::nullopt_t
- class std::experimental::fundamentals_v1::optional< _Tp >

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **__cpp_lib_experimental_optional**
- #define **_GLIBCXX_EXPERIMENTAL_OPTIONAL**

Variables

- constexpr in_place_t std::experimental::in_place
- constexpr nullopt_t std::experimental::nullopt

6.600.1 Detailed Description

This is a TS C++ Library header.

6.601 optional File Reference

Macros

- #define **__glibcxx_want_constrained_equality**
- #define **__glibcxx_want_freestanding_optional**
- #define **__glibcxx_want_optional**
- #define **__glibcxx_want_optional_range_support**
- #define **_GLIBCXX_OPTIONAL**

6.601.1 Detailed Description

This is a Standard C++ Library header.

6.602 ostream File Reference

Namespaces

- namespace std

Macros

- #define **__glibcxx_want_print**
- #define **_GLIBCXX_OSTREAM**

Functions

- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & std::endl (basic_ostream< _CharT, _Traits > &__os)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & std::ends (basic_ostream< _CharT, _Traits > &__os)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & std::flush (basic_ostream< _CharT, _Traits > &__os)

6.602.1 Detailed Description

This is a Standard C++ Library header.

6.603 algo.h File Reference

Classes

- struct std::__parallel::_CRandNumber< _MustBeInt >

Namespaces

- namespace std
- namespace std::__parallel

Functions

- template<typename _FIterator, typename _BinaryPredicate, typename _IteratorTag>
  _FIterator **std::__parallel::__adjacent_find_switch** (_FIterator __begin, _FIterator __end, _BinaryPredicate ↩
  __pred, _IteratorTag)
- template<typename _FIterator, typename _IteratorTag>
  _FIterator **std::__parallel::__adjacent_find_switch** (_FIterator __begin, _FIterator __end, _IteratorTag)
- template<typename _RAIter, typename _BinaryPredicate>
  _RAIter **std::__parallel::__adjacent_find_switch** (_RAIter __begin, _RAIter __end, _BinaryPredicate __pred, random_access_iterator_tag)
- template<typename _RAIter>
  _RAIter **std::__parallel::__adjacent_find_switch** (_RAIter __begin, _RAIter __end, random_access_iterator_tag)
- template<typename _IIter, typename _Predicate, typename _IteratorTag>
  iterator_traits< _IIter >::difference_type **std::__parallel::__count_if_switch** (_IIter __begin, _IIter __end, _↩
  Predicate __pred, _IteratorTag)
- template<typename _RAIter, typename _Predicate>
  iterator_traits< _RAIter >::difference_type **std::__parallel::__count_if_switch** (_RAIter __begin, _RAIter __↩
  end, _Predicate __pred, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter, typename _Tp, typename _IteratorTag>
  iterator_traits< _IIter >::difference_type **std::__parallel::__count_switch** (_IIter __begin, _IIter __end, const _Tp &__value, _IteratorTag)
- template<typename _RAIter, typename _Tp>
  iterator_traits< _RAIter >::difference_type **std::__parallel::__count_switch** (_RAIter __begin, _RAIter __end, const _Tp &__value, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)

- template<typename _IIter, typename _FIterator, typename _BinaryPredicate, typename _IteratorTag1, typename _IteratorTag2>
  _IIter **std::__parallel::__find_first_of_switch** (_IIter __begin1, _IIter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, _IteratorTag1, _IteratorTag2)
- template<typename _IIter, typename _FIterator, typename _IteratorTag1, typename _IteratorTag2>
  _IIter **std::__parallel::__find_first_of_switch** (_IIter __begin1, _IIter __end1, _FIterator __begin2, _FIterator __end2, _IteratorTag1, _IteratorTag2)
- template<typename _RAIter, typename _FIterator, typename _BinaryPredicate, typename _IteratorTag>
  _RAIter **std::__parallel::__find_first_of_switch** (_RAIter __begin1, _RAIter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, random_access_iterator_tag, _IteratorTag)
- template<typename _IIter, typename _Predicate, typename _IteratorTag>
  _IIter **std::__parallel::__find_if_switch** (_IIter __begin, _IIter __end, _Predicate __pred, _IteratorTag)
- template<typename _RAIter, typename _Predicate>
  _RAIter **std::__parallel::__find_if_switch** (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)
- template<typename _IIter, typename _Tp, typename _IteratorTag>
  _IIter **std::__parallel::__find_switch** (_IIter __begin, _IIter __end, const _Tp &__val, _IteratorTag)
- template<typename _RAIter, typename _Tp>
  _RAIter **std::__parallel::__find_switch** (_RAIter __begin, _RAIter __end, const _Tp &__val, random_access_iterator_tag)
- template<typename _IIter, typename _Function, typename _IteratorTag>
  _Function **std::__parallel::__for_each_switch** (_IIter __begin, _IIter __end, _Function __f, _IteratorTag)
- template<typename _RAIter, typename _Function>
  _Function **std::__parallel::__for_each_switch** (_RAIter __begin, _RAIter __end, _Function __f, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _OutputIterator, typename _Size, typename _Generator, typename _IteratorTag>
  _OutputIterator **std::__parallel::__generate_n_switch** (_OutputIterator __begin, _Size __n, _Generator __gen, _IteratorTag)
- template<typename _RAIter, typename _Size, typename _Generator>
  _RAIter **std::__parallel::__generate_n_switch** (_RAIter __begin, _Size __n, _Generator __gen, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator, typename _Generator, typename _IteratorTag>
  void **std::__parallel::__generate_switch** (_FIterator __begin, _FIterator __end, _Generator __gen, _IteratorTag)
- template<typename _RAIter, typename _Generator>
  void **std::__parallel::__generate_switch** (_RAIter __begin, _RAIter __end, _Generator __gen, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator, typename _Compare, typename _IteratorTag>
  _FIterator **std::__parallel::__max_element_switch** (_FIterator __begin, _FIterator __end, _Compare __comp, _IteratorTag)
- template<typename _RAIter, typename _Compare>
  _RAIter **std::__parallel::__max_element_switch** (_RAIter __begin, _RAIter __end, _Compare __comp, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Compare, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3>
  _OutputIterator **std::__parallel::__merge_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Compare __comp, _IteratorTag1, _IteratorTag2, _IteratorTag3)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Compare>
  _OutputIterator **std::__parallel::__merge_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Compare __comp, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _FIterator, typename _Compare, typename _IteratorTag>
  _FIterator **std::__parallel::__min_element_switch** (_FIterator __begin, _FIterator __end, _Compare __comp, _IteratorTag)
- template<typename _RAIter, typename _Compare>
  _RAIter **std::__parallel::__min_element_switch** (_RAIter __begin, _RAIter __end, _Compare __comp, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)

- template<typename _FIterator, typename _Predicate, typename _IteratorTag>
  _FIterator **std::__parallel::__partition_switch** (_FIterator __begin, _FIterator __end, _Predicate __pred, _IteratorTag)
- template<typename _RAIter, typename _Predicate>
  _RAIter **std::__parallel::__partition_switch** (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)
- template<typename _FIterator, typename _Predicate, typename _Tp, typename _IteratorTag>
  void **std::__parallel::__replace_if_switch** (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value, _IteratorTag)
- template<typename _RAIter, typename _Predicate, typename _Tp>
  void **std::__parallel::__replace_if_switch** (_RAIter __begin, _RAIter __end, _Predicate __pred, const _Tp &__new_value, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator, typename _Tp, typename _IteratorTag>
  void **std::__parallel::__replace_switch** (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value, _IteratorTag)
- template<typename _RAIter, typename _Tp>
  void **std::__parallel::__replace_switch** (_RAIter __begin, _RAIter __end, const _Tp &__old_value, const _Tp &__new_value, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate, typename _IteratorTag>
  _FIterator **std::__parallel::__search_n_switch** (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, _IteratorTag)
- template<typename _RAIter, typename _Integer, typename _Tp, typename _BinaryPredicate>
  _RAIter **std::__parallel::__search_n_switch** (_RAIter __begin, _RAIter __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, random_access_iterator_tag)
- template<typename _FIterator1, typename _FIterator2, typename _IteratorTag1, typename _IteratorTag2>
  _FIterator1 **std::__parallel::__search_switch** (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _IteratorTag1, _IteratorTag2)
- template<typename _RAIter1, typename _RAIter2>
  _RAIter1 **std::__parallel::__search_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3>
  _OutputIterator **std::__parallel::__set_difference_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)
- template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate>
  _Output_RAIter **std::__parallel::__set_difference_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3>
  _OutputIterator **std::__parallel::__set_intersection_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)
- template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate>
  _Output_RAIter **std::__parallel::__set_intersection_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3>
  _OutputIterator **std::__parallel::__set_symmetric_difference_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)
- template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate>
  _Output_RAIter **std::__parallel::__set_symmetric_difference_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)

- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3>
  _OutputIterator **std::__parallel::__set_union_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)
- template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate>
  _Output_RAIter **std::__parallel::__set_union_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation, typename _IteratorTag1, typename _IteratorTag2>
  _RAIter2 **std::__parallel::__transform1_switch** (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __unary_op, _IteratorTag1, _IteratorTag2)
- template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation>
  _RAIter2 **std::__parallel::__transform1_switch** (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __unary_op, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _BinaryOperation, typename _Tag1, typename _Tag2, typename _Tag3>
  _OutputIterator **std::__parallel::__transform2_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, _Tag1, _Tag2, _Tag3)
- template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BinaryOperation>
  _RAIter3 **std::__parallel::__transform2_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter3 __result, _BinaryOperation __binary_op, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter, typename _OutputIterator, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2>
  _OutputIterator **std::__parallel::__unique_copy_switch** (_IIter __begin, _IIter __last, _OutputIterator __out, _Predicate __pred, _IteratorTag1, _IteratorTag2)
- template<typename _RAIter, typename _RandomAccessOutputIterator, typename _Predicate>
  _RandomAccessOutputIterator **std::__parallel::__unique_copy_switch** (_RAIter __begin, _RAIter __last, _RandomAccessOutputIterator __out, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _FIterator>
  _FIterator **std::__parallel::adjacent_find** (_FIterator __begin, _FIterator __end)
- template<typename _FIterator>
  _FIterator **std::__parallel::adjacent_find** (_FIterator __begin, _FIterator __end, __gnu_parallel::sequential_tag)
- template<typename _FIterator, typename _BinaryPredicate>
  _FIterator **std::__parallel::adjacent_find** (_FIterator __begin, _FIterator __end, _BinaryPredicate __binary_pred, __gnu_parallel::sequential_tag)
- template<typename _FIterator, typename _BinaryPredicate>
  _FIterator **std::__parallel::adjacent_find** (_FIterator __begin, _FIterator __end, _BinaryPredicate __pred)
- template<typename _IIter, typename _Tp>
  iterator_traits< _IIter >::difference_type **std::__parallel::count** (_IIter __begin, _IIter __end, const _Tp &__value)
- template<typename _IIter, typename _Tp>
  iterator_traits< _IIter >::difference_type **std::__parallel::count** (_IIter __begin, _IIter __end, const _Tp &__value, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter, typename _Tp>
  iterator_traits< _IIter >::difference_type **std::__parallel::count** (_IIter __begin, _IIter __end, const _Tp &__value, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _Predicate>
  iterator_traits< _IIter >::difference_type **std::__parallel::count_if** (_IIter __begin, _IIter __end, _Predicate __pred)
- template<typename _IIter, typename _Predicate>
  iterator_traits< _IIter >::difference_type **std::__parallel::count_if** (_IIter __begin, _IIter __end, _Predicate __pred, __gnu_parallel::_Parallelism __parallelism_tag)

- template<typename _IIter, typename _Predicate>
  iterator_traits< _IIter >::difference_type **std::__parallel::count_if** (_IIter __begin, _IIter __end, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _Tp>
  _IIter **std::__parallel::find** (_IIter __begin, _IIter __end, const _Tp &__val)
- template<typename _IIter, typename _Tp>
  _IIter **std::__parallel::find** (_IIter __begin, _IIter __end, const _Tp &__val, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _FIterator>
  _IIter **std::__parallel::find_first_of** (_IIter __begin1, _IIter __end1, _FIterator __begin2, _FIterator __end2)
- template<typename _IIter, typename _FIterator>
  _IIter **std::__parallel::find_first_of** (_IIter __begin1, _IIter __end1, _FIterator __begin2, _FIterator __end2, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _FIterator, typename _BinaryPredicate>
  _IIter **std::__parallel::find_first_of** (_IIter __begin1, _IIter __end1, _FIterator __begin2, _FIterator __end2, BinaryPredicate __comp)
- template<typename _IIter, typename _FIterator, typename _BinaryPredicate>
  _IIter **std::__parallel::find_first_of** (_IIter __begin1, _IIter __end1, _FIterator __begin2, _FIterator __end2, BinaryPredicate __comp, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _Predicate>
  _IIter **std::__parallel::find_if** (_IIter __begin, _IIter __end, _Predicate __pred)
- template<typename _IIter, typename _Predicate>
  _IIter **std::__parallel::find_if** (_IIter __begin, _IIter __end, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _Function>
  _Function **std::__parallel::for_each** (_IIter __begin, _IIter __end, _Function __f, __gnu_parallel::sequential_tag)
- template<typename _Iterator, typename _Function>
  _Function **std::__parallel::for_each** (_Iterator __begin, _Iterator __end, _Function __f)
- template<typename _Iterator, typename _Function>
  _Function **std::__parallel::for_each** (_Iterator __begin, _Iterator __end, _Function __f, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator, typename _Generator>
  void **std::__parallel::generate** (_FIterator __begin, _FIterator __end, _Generator __gen)
- template<typename _FIterator, typename _Generator>
  void **std::__parallel::generate** (_FIterator __begin, _FIterator __end, _Generator __gen, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator, typename _Generator>
  void **std::__parallel::generate** (_FIterator __begin, _FIterator __end, _Generator __gen, __gnu_parallel::sequential_tag)
- template<typename _OutputIterator, typename _Size, typename _Generator>
  _OutputIterator **std::__parallel::generate_n** (_OutputIterator __begin, _Size __n, _Generator __gen)
- template<typename _OutputIterator, typename _Size, typename _Generator>
  _OutputIterator **std::__parallel::generate_n** (_OutputIterator __begin, _Size __n, _Generator __gen, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _OutputIterator, typename _Size, typename _Generator>
  _OutputIterator **std::__parallel::generate_n** (_OutputIterator __begin, _Size __n, _Generator __gen, __gnu_parallel::sequential_tag)
- template<typename _FIterator>
  _FIterator **std::__parallel::max_element** (_FIterator __begin, _FIterator __end)
- template<typename _FIterator>
  _FIterator **std::__parallel::max_element** (_FIterator __begin, _FIterator __end, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator>
  _FIterator **std::__parallel::max_element** (_FIterator __begin, _FIterator __end, __gnu_parallel::sequential_tag)
- template<typename _FIterator, typename _Compare>
  _FIterator **std::__parallel::max_element** (_FIterator __begin, _FIterator __end, _Compare __comp)

- template<typename _FIterator, typename _Compare>
  _FIterator **std::__parallel::max_element** (_FIterator __begin, _FIterator __end, _Compare __comp, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator, typename _Compare>
  _FIterator **std::__parallel::max_element** (_FIterator __begin, _FIterator __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator>
  _OutputIterator **std::__parallel::merge** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator>
  _OutputIterator **std::__parallel::merge** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Compare>
  _OutputIterator **std::__parallel::merge** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Compare __comp)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Compare>
  _OutputIterator **std::__parallel::merge** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _FIterator>
  _FIterator **std::__parallel::min_element** (_FIterator __begin, _FIterator __end)
- template<typename _FIterator>
  _FIterator **std::__parallel::min_element** (_FIterator __begin, _FIterator __end, __gnu_parallel::_Parallelism _parallelism_tag)
- template<typename _FIterator>
  _FIterator **std::__parallel::min_element** (_FIterator __begin, _FIterator __end, __gnu_parallel::sequential_tag)
- template<typename _FIterator, typename _Compare>
  _FIterator **std::__parallel::min_element** (_FIterator __begin, _FIterator __end, _Compare __comp)
- template<typename _FIterator, typename _Compare>
  _FIterator **std::__parallel::min_element** (_FIterator __begin, _FIterator __end, _Compare __comp, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator, typename _Compare>
  _FIterator **std::__parallel::min_element** (_FIterator __begin, _FIterator __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIter>
  void **std::__parallel::nth_element** (_RAIter __begin, _RAIter __nth, _RAIter __end)
- template<typename _RAIter>
  void **std::__parallel::nth_element** (_RAIter __begin, _RAIter __nth, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter, typename _Compare>
  void **std::__parallel::nth_element** (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)
- template<typename _RAIter, typename _Compare>
  void **std::__parallel::nth_element** (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIter>
  void **std::__parallel::partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end)
- template<typename _RAIter>
  void **std::__parallel::partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter, typename _Compare>
  void **std::__parallel::partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)
- template<typename _RAIter, typename _Compare>
  void **std::__parallel::partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _FIterator, typename _Predicate>
  _FIterator **std::__parallel::partition** (_FIterator __begin, _FIterator __end, _Predicate __pred)

- template<typename _FIterator, typename _Predicate>
  _FIterator **std::__parallel::partition** (_FIterator __begin, _FIterator __end, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _RAIter>
  void **std::__parallel::random_shuffle** (_RAIter __begin, _RAIter __end)
- template<typename _RAIter>
  void **std::__parallel::random_shuffle** (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter, typename _RandomNumberGenerator>
  void **std::__parallel::random_shuffle** (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &&__rand)
- template<typename _RAIter, typename _RandomNumberGenerator>
  void **std::__parallel::random_shuffle** (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &__rand, __gnu_parallel::sequential_tag)
- template<typename _FIterator, typename _Tp>
  void **std::__parallel::replace** (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__↩ new_value)
- template<typename _FIterator, typename _Tp>
  void **std::__parallel::replace** (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__↩ new_value, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator, typename _Tp>
  void **std::__parallel::replace** (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__↩ new_value, __gnu_parallel::sequential_tag)
- template<typename _FIterator, typename _Predicate, typename _Tp>
  void **std::__parallel::replace_if** (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new↩ _value)
- template<typename _FIterator, typename _Predicate, typename _Tp>
  void **std::__parallel::replace_if** (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new↩ _value, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator, typename _Predicate, typename _Tp>
  void **std::__parallel::replace_if** (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new↩ _value, __gnu_parallel::sequential_tag)
- template<typename _FIterator1, typename _FIterator2>
  _FIterator1 **std::__parallel::search** (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2)
- template<typename _FIterator1, typename _FIterator2>
  _FIterator1 **std::__parallel::search** (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, __gnu_parallel::sequential_tag)
- template<typename _ForwardIterator, typename _Searcher>
  _ForwardIterator std::__parallel::search (_ForwardIterator __first, _ForwardIterator __last, const _Searcher &_↩ _searcher)
- template<typename _FIterator, typename _Integer, typename _Tp>
  _FIterator **std::__parallel::search_n** (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val)
- template<typename _FIterator, typename _Integer, typename _Tp>
  _FIterator **std::__parallel::search_n** (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, __gnu_parallel::sequential_tag)
- template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate>
  _FIterator **std::__parallel::search_n** (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)
- template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate>
  _FIterator **std::__parallel::search_n** (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator>
  _OutputIterator **std::__parallel::set_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __↩ end2, _OutputIterator __out)

- template<typename _IIter1, typename _IIter2, typename _OutputIterator>
  _OutputIterator **std::__parallel::set_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __↩ end2, _OutputIterator __out, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate>
  _OutputIterator **std::__parallel::set_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __↩ end2, _OutputIterator __out, _Predicate __pred)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate>
  _OutputIterator **std::__parallel::set_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __↩ end2, _OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator>
  _OutputIterator **std::__parallel::set_intersection** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 ↩ __end2, _OutputIterator __out)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator>
  _OutputIterator **std::__parallel::set_intersection** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 ↩ __end2, _OutputIterator __out, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate>
  _OutputIterator **std::__parallel::set_intersection** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 ↩ __end2, _OutputIterator __out, _Predicate __pred)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate>
  _OutputIterator **std::__parallel::set_intersection** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 ↩ __end2, _OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator>
  _OutputIterator **std::__parallel::set_symmetric_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator>
  _OutputIterator **std::__parallel::set_symmetric_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate>
  _OutputIterator **std::__parallel::set_symmetric_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate>
  _OutputIterator **std::__parallel::set_symmetric_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator>
  _OutputIterator **std::__parallel::set_union** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator>
  _OutputIterator **std::__parallel::set_union** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate>
  _OutputIterator **std::__parallel::set_union** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate>
  _OutputIterator **std::__parallel::set_union** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _RAIter>
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end)
- template<typename _RAIter>
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::balanced_quicksort_tag __↩ parallelism)
- template<typename _RAIter>
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::default_parallel_tag __parallelism)

- template<typename _RAIter>
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_exact_tag _↩
  _parallelism)
- template<typename _RAIter>
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_sampling_tag
  __parallelism)
- template<typename _RAIter>
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_tag __↩
  parallelism)
- template<typename _RAIter>
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::parallel_tag __parallelism)
- template<typename _RAIter>
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::quicksort_tag __parallelism)
- template<typename _RAIter>
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter, typename _Compare>
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end, _Compare __comp)
- template<typename _RAIter, typename _Compare>
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIter, typename _Compare, typename _Parallelism>
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)
- template<typename _RAIter>
  void **std::__parallel::stable_sort** (_RAIter __begin, _RAIter __end)
- template<typename _RAIter>
  void **std::__parallel::stable_sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::balanced_quicksort_tag _↩
  _parallelism)
- template<typename _RAIter>
  void **std::__parallel::stable_sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::default_parallel_tag __↩
  parallelism)
- template<typename _RAIter>
  void **std::__parallel::stable_sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_tag ↩
  __parallelism)
- template<typename _RAIter>
  void **std::__parallel::stable_sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::parallel_tag __parallelism)
- template<typename _RAIter>
  void **std::__parallel::stable_sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::quicksort_tag __parallelism)
- template<typename _RAIter>
  void **std::__parallel::stable_sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter, typename _Compare>
  void **std::__parallel::stable_sort** (_RAIter __begin, _RAIter __end, _Compare __comp)
- template<typename _RAIter, typename _Compare>
  void **std::__parallel::stable_sort** (_RAIter __begin, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIter, typename _Compare, typename _Parallelism>
  void **std::__parallel::stable_sort** (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __↩
  parallelism)
- template<typename _IIter, typename _OutputIterator, typename _UnaryOperation>
  _OutputIterator **std::__parallel::transform** (_IIter __begin, _IIter __end, _OutputIterator __result, _Unary↩
  Operation __unary_op)
- template<typename _IIter, typename _OutputIterator, typename _UnaryOperation>
  _OutputIterator **std::__parallel::transform** (_IIter __begin, _IIter __end, _OutputIterator __result, _Unary↩
  Operation __unary_op, __gnu_parallel::_Parallelism __parallelism_tag)

- template<typename _IIter, typename _OutputIterator, typename _UnaryOperation>
  _OutputIterator **std::__parallel::transform** (_IIter __begin, _IIter __end, _OutputIterator __result, _Unary↩
  Operation __unary_op, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator **std::__parallel::transform** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator **std::__parallel::transform** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator **std::__parallel::transform** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OutputIterator>
  _OutputIterator **std::__parallel::unique_copy** (_IIter __begin1, _IIter __end1, _OutputIterator __out)
- template<typename _IIter, typename _OutputIterator>
  _OutputIterator **std::__parallel::unique_copy** (_IIter __begin1, _IIter __end1, _OutputIterator __out, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OutputIterator, typename _Predicate>
  _OutputIterator **std::__parallel::unique_copy** (_IIter __begin1, _IIter __end1, _OutputIterator __out, _Predicate __pred)
- template<typename _IIter, typename _OutputIterator, typename _Predicate>
  _OutputIterator **std::__parallel::unique_copy** (_IIter __begin1, _IIter __end1, _OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)

6.603.1 Detailed Description

Parallel STL function calls corresponding to the stl_algo.h header.
The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

6.604 algobase.h File Reference

Namespaces

- namespace std
- namespace std::__parallel

Functions

- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2>
  bool **std::__parallel::__equal_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _↩
  Predicate __pred, _IteratorTag1, _IteratorTag2)
- template<typename _RAIter1, typename _RAIter2, typename _Predicate>
  bool **std::__parallel::__equal_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 _↩
  _end2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2>
  bool **std::__parallel::__lexicographical_compare_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Predicate __pred, _IteratorTag1, _IteratorTag2)
- template<typename _RAIter1, typename _RAIter2, typename _Predicate>
  bool **std::__parallel::__lexicographical_compare_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 _↩
  _begin2, _RAIter2 __end2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2>
  pair< _IIter1, _IIter2 > **std::__parallel::__mismatch_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Predicate __pred, _IteratorTag1, _IteratorTag2)

- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2>
  pair< _IIter1, _IIter2 > **std::__parallel::__mismatch_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred, _IteratorTag1, _IteratorTag2)
- template<typename _RAIter1, typename _RAIter2, typename _Predicate>
  pair< _RAIter1, _RAIter2 > **std::__parallel::__mismatch_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _RAIter1, typename _RAIter2, typename _Predicate>
  pair< _RAIter1, _RAIter2 > **std::__parallel::__mismatch_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate, typename _IteratorTag1, typename _IteratorTag2>
  _FIterator1 **std::__parallel::__search_switch** (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred, _IteratorTag1, _IteratorTag2)
- template<typename _RAIter1, typename _RAIter2, typename _BinaryPredicate>
  _RAIter1 **std::__parallel::__search_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _BinaryPredicate __pred, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter1, typename _IIter2>
  constexpr bool **std::__parallel::equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2)
- template<typename _IIter1, typename _IIter2>
  bool **std::__parallel::equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2>
  constexpr bool **std::__parallel::equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2)
- template<typename _IIter1, typename _IIter2>
  bool **std::__parallel::equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _BinaryPredicate>
  constexpr bool **std::__parallel::equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _BinaryPredicate __binary_pred)
- template<typename _IIter1, typename _IIter2, typename _BinaryPredicate>
  bool **std::__parallel::equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _BinaryPredicate __binary_pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate>
  constexpr bool **std::__parallel::equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred)
- template<typename _IIter1, typename _IIter2, typename _Predicate>
  bool **std::__parallel::equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2>
  constexpr bool **std::__parallel::lexicographical_compare** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2)
- template<typename _IIter1, typename _IIter2>
  bool **std::__parallel::lexicographical_compare** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate>
  constexpr bool **std::__parallel::lexicographical_compare** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Predicate __pred)
- template<typename _IIter1, typename _IIter2, typename _Predicate>
  bool **std::__parallel::lexicographical_compare** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2>
  pair< _IIter1, _IIter2 > **std::__parallel::mismatch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2)
- template<typename _IIter1, typename _IIter2>
  pair< _IIter1, _IIter2 > **std::__parallel::mismatch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2>
  pair< _IIter1, _IIter2 > **std::__parallel::mismatch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2)

- template<typename _IIter1, typename _IIter2, typename _Predicate>
  pair< _IIter1, _IIter2 > **std::__parallel::mismatch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred)
- template<typename _IIter1, typename _IIter2, typename _Predicate>
  pair< _IIter1, _IIter2 > **std::__parallel::mismatch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>
  pair< _InputIterator1, _InputIterator2 > **std::__parallel::mismatch** (_InputIterator1 __begin1, _InputIterator1 __end1, _InputIterator2 __begin2, _InputIterator2 __end2, _BinaryPredicate __binary_pred)
- template<typename _InputIterator1, typename _InputIterator2>
  pair< _InputIterator1, _InputIterator2 > **std::__parallel::mismatch** (_InputIterator1 __first1, _InputIterator1 _last1, _InputIterator2 __first2, _InputIterator2 __last2, __gnu_parallel::sequential_tag)
- template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>
  pair< _InputIterator1, _InputIterator2 > **std::__parallel::mismatch** (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _BinaryPredicate __binary_pred, __gnu_parallel::sequential_tag)
- template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate>
  constexpr _FIterator1 **std::__parallel::search** (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred)
- template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate>
  _FIterator1 **std::__parallel::search** (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred, __gnu_parallel::sequential_tag)

6.604.1 Detailed Description

Parallel STL function calls corresponding to the stl_algobase.h header. The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

6.605 balanced_quicksort.h File Reference

Classes

- struct __gnu_parallel::_QSBThreadLocal< _RAIter >

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _RAIter, typename _Compare>
  void __gnu_parallel::__parallel_sort_qsb (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)
- template<typename _RAIter, typename _Compare>
  void __gnu_parallel::__qsb_conquer (_QSBThreadLocal< _RAIter > **__tls, _RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __iam, _ThreadIndex __num_threads, bool __parent_wait)
- template<typename _RAIter, typename _Compare>
  std::iterator_traits< _RAIter >::difference_type __gnu_parallel::__qsb_divide (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)
- template<typename _RAIter, typename _Compare>
  void __gnu_parallel::__qsb_local_sort_with_helping (_QSBThreadLocal< _RAIter > **__tls, _Compare &__comp, _ThreadIndex __iam, bool __wait)

6.605.1 Detailed Description

Implementation of a dynamically load-balanced parallel quicksort.
It works in-place and needs only logarithmic extra memory. The algorithm is similar to the one proposed in
P. Tsigas and Y. Zhang. A simple, fast parallel implementation of quicksort and its performance evaluation on SUN enterprise 10000. In 11th Euromicro Conference on Parallel, Distributed and Network-Based Processing, page 372, 2003.
This file is a GNU parallel extension to the Standard C++ Library.

6.606 base.h File Reference

Classes

- class __gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >
- class __gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >
- class __gnu_parallel::__unary_negate< _Predicate, argument_type >
- class __gnu_parallel::_EqualFromLess< _T1, _T2, _Compare >
- struct __gnu_parallel::_EqualTo< _T1, _T2 >
- struct __gnu_parallel::_Less< _T1, _T2 >
- struct __gnu_parallel::_Multiplies< _Tp1, _Tp2, _Result >
- struct __gnu_parallel::_Plus< _Tp1, _Tp2, _Result >
- class __gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >
- class __gnu_parallel::_PseudoSequenceIterator< _Tp, _DifferenceTp >

Namespaces

- namespace __gnu_parallel
- namespace __gnu_sequential
- namespace std
- namespace std::__parallel

Macros

- #define **_GLIBCXX_PARALLEL_ASSERT**(_Condition)

Functions

- void __gnu_parallel::__decode2 (_CASable __x, int &__a, int &__b)
- _CASable __gnu_parallel::__encode2 (int __a, int __b)
- _ThreadIndex **__gnu_parallel::__get_max_threads** ()
- bool **__gnu_parallel::__is_parallel** (const _Parallelism __p)
- template<typename _RAIter, typename _Compare>
  _RAIter __gnu_parallel::__median_of_three_iterators (_RAIter __a, _RAIter __b, _RAIter __c, _Compare __comp)
- template<typename _Size>
  _Size __gnu_parallel::__rd_log2 (_Size __n)
- template<typename _Tp>
  const _Tp & __gnu_parallel::max (const _Tp &__a, const _Tp &__b)
- template<typename _Tp>
  const _Tp & __gnu_parallel::min (const _Tp &__a, const _Tp &__b)

6.606.1 Detailed Description

Sequential helper functions. This file is a GNU parallel extension to the Standard C++ Library.

6.607 basic_iterator.h File Reference

6.607.1 Detailed Description

Includes the original header files concerned with iterators except for stream iterators. This file is a GNU parallel extension to the Standard C++ Library.

6.608 checkers.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _IIter, typename _Compare>
  bool __gnu_parallel::__is_sorted (_IIter __begin, _IIter __end, _Compare __comp)

6.608.1 Detailed Description

Routines for checking the correctness of algorithm results. This file is a GNU parallel extension to the Standard C++ Library.

6.609 compiletime_settings.h File Reference

Macros

- #define _GLIBCXX_CALL(__n)
- #define _GLIBCXX_PARALLEL_ASSERTIONS
- #define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_L1
- #define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_TLB
- #define _GLIBCXX_SCALE_DOWN_FPU
- #define _GLIBCXX_VERBOSE_LEVEL

6.609.1 Detailed Description

Defines on options concerning debugging and performance, at compile-time. This file is a GNU parallel extension to the Standard C++ Library.

6.609.2 Macro Definition Documentation

_GLIBCXX_CALL

```
#define _GLIBCXX_CALL(
                __n)
```

Macro to produce log message when entering a function.

Parameters

_↩ _n	Input size.

See also

_GLIBCXX_VERBOSE_LEVEL

Referenced by __gnu_parallel::__find_template(), __gnu_parallel::__find_template(), __gnu_parallel::__find_template(), __gnu_parallel::__for_each_template_random_access_workstealing(), __gnu_parallel::__merge_advance(), __gnu_parallel::__parallel_n __gnu_parallel::__parallel_partial_sum(), __gnu_parallel::__parallel_partition(), __gnu_parallel::__parallel_random_shuffle_drs(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort_qs() __gnu_parallel::__parallel_sort_qsb(), __gnu_parallel::__parallel_unique_copy(), __gnu_parallel::__search_template(), __gnu_parallel::__sequential_multiway_merge(), __gnu_parallel::multiseq_partition(), __gnu_parallel::multiseq_selection(), __gnu_parallel::multiway_merge(), __gnu_parallel::multiway_merge_3_variant(), __gnu_parallel::multiway_merge_4_variant(), __gnu_parallel::multiway_merge_loser_tree(), __gnu_parallel::multiway_merge_loser_tree_sentinel(), __gnu_parallel::multiway_merge_lo __gnu_parallel::multiway_merge_sentinels(), __gnu_parallel::parallel_multiway_merge(), and __gnu_parallel::parallel_sort_mwms().

_GLIBCXX_PARALLEL_ASSERTIONS

```
#define _GLIBCXX_PARALLEL_ASSERTIONS
```

Switch on many _GLIBCXX_PARALLEL_ASSERTions in parallel code. Should be switched on only locally.
Referenced by __gnu_parallel::__qsb_local_sort_with_helping().

_GLIBCXX_RANDOM_SHUFFLE_CONSIDER_L1

```
#define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_L1
```

Switch on many _GLIBCXX_PARALLEL_ASSERTions in parallel code. Consider the size of the L1 cache for gnu_↩ parallel::__parallel_random_shuffle().

_GLIBCXX_RANDOM_SHUFFLE_CONSIDER_TLB

```
#define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_TLB
```

Switch on many _GLIBCXX_PARALLEL_ASSERTions in parallel code. Consider the size of the TLB for gnu_parallel↩ ::__parallel_random_shuffle().

_GLIBCXX_SCALE_DOWN_FPU

```
#define _GLIBCXX_SCALE_DOWN_FPU
```

Use floating-point scaling instead of modulo for mapping random numbers to a range. This can be faster on certain CPUs.

_GLIBCXX_VERBOSE_LEVEL

```
#define _GLIBCXX_VERBOSE_LEVEL
```

Determine verbosity level of the parallel mode. Level 1 prints a message each time a parallel-mode function is entered.

6.610 equally_split.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _DifferenceType, typename _OutputIterator>
  _OutputIterator __gnu_parallel::__equally_split (_DifferenceType __n, _ThreadIndex __num_threads, _Output↩ Iterator __s)

- template<typename _DifferenceType>
  _DifferenceType __gnu_parallel::__equally_split_point (_DifferenceType __n, _ThreadIndex __num_threads, _ThreadIndex __thread_no)

6.610.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

6.611 features.h File Reference

Macros

- #define _GLIBCXX_BAL_QUICKSORT
- #define _GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS
- #define _GLIBCXX_FIND_EQUAL_SPLIT
- #define _GLIBCXX_FIND_GROWING_BLOCKS
- #define _GLIBCXX_MERGESORT
- #define _GLIBCXX_QUICKSORT
- #define _GLIBCXX_TREE_DYNAMIC_BALANCING
- #define _GLIBCXX_TREE_FULL_COPY
- #define _GLIBCXX_TREE_INITIAL_SPLITTING

6.611.1 Detailed Description

Defines on whether to include algorithm variants.
Less variants reduce executable size and compile time. This file is a GNU parallel extension to the Standard C++ Library.

6.611.2 Macro Definition Documentation

_GLIBCXX_BAL_QUICKSORT

```
#define _GLIBCXX_BAL_QUICKSORT
```

Include parallel dynamically load-balanced quicksort.

See also

__gnu_parallel::_Settings::sort_algorithm

_GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS

```
#define _GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS
```

Include the equal-sized blocks variant for std::find.

See also

__gnu_parallel::_Settings::find_algorithm

_GLIBCXX_FIND_EQUAL_SPLIT

```
#define _GLIBCXX_FIND_EQUAL_SPLIT
```

Include the equal splitting variant for std::find.

See also

__gnu_parallel::_Settings::find_algorithm

_GLIBCXX_FIND_GROWING_BLOCKS

```
#define _GLIBCXX_FIND_GROWING_BLOCKS
```

Include the growing blocks variant for std::find.

See also

__gnu_parallel::_Settings::find_algorithm

_GLIBCXX_MERGESORT

```
#define _GLIBCXX_MERGESORT
```

Include parallel multi-way mergesort.

See also

__gnu_parallel::_Settings::sort_algorithm

_GLIBCXX_QUICKSORT

```
#define _GLIBCXX_QUICKSORT
```

Include parallel unbalanced quicksort.

See also

__gnu_parallel::_Settings::sort_algorithm

_GLIBCXX_TREE_DYNAMIC_BALANCING

```
#define _GLIBCXX_TREE_DYNAMIC_BALANCING
```

Include the dynamic balancing variant for _Rb_tree::insert_unique(_IIter beg, _IIter __end).

See also

__gnu_parallel::_Rb_tree

_GLIBCXX_TREE_FULL_COPY

```
#define _GLIBCXX_TREE_FULL_COPY
```

In order to sort the input sequence of _Rb_tree::insert_unique(_IIter beg, _IIter __end) a full copy of the input elements is done.

See also

__gnu_parallel::_Rb_tree

_GLIBCXX_TREE_INITIAL_SPLITTING

```
#define _GLIBCXX_TREE_INITIAL_SPLITTING
```

Include the initial splitting variant for _Rb_tree::insert_unique(_IIter beg, _IIter __end).

See also

__gnu_parallel::_Rb_tree

6.612 find.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector>
  std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector)
- template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector>
  std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, constant_size_blocks_tag)
- template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector>
  std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, equal_split_tag)
- template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector>
  std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, growing_blocks_tag)

6.612.1 Detailed Description

Parallel implementation base for std::find(), std::equal() and related functions. This file is a GNU parallel extension to the Standard C++ Library.

6.613 find_selectors.h File Reference

Classes

- struct __gnu_parallel::__adjacent_find_selector
- struct __gnu_parallel::__find_first_of_selector< _FIterator >
- struct __gnu_parallel::__find_if_selector
- struct __gnu_parallel::__generic_find_selector
- struct __gnu_parallel::__mismatch_selector

Namespaces

- namespace __gnu_parallel

6.613.1 Detailed Description

_Function objects representing different tasks to be plugged into the parallel find algorithm. This file is a GNU parallel extension to the Standard C++ Library.

6.614 for_each.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _IIter, typename _UserOp, typename _Functionality, typename _Red, typename _Result>
  _UserOp __gnu_parallel::__for_each_template_random_access (_IIter __begin, _IIter __end, _UserOp __user_op, _Functionality &__functionality, _Red __reduction, _Result __reduction_start, _Result &__output, typename std::iterator_traits< _IIter >::difference_type __bound, _Parallelism __parallelism_tag)

6.614.1 Detailed Description

Main interface for embarrassingly parallel functions.
The explicit implementation are in other header files, like workstealing.h, par_loop.h, omp_loop.h, and omp_loop_↩ static.h. This file is a GNU parallel extension to the Standard C++ Library.

6.615 for_each_selectors.h File Reference

Classes

- struct __gnu_parallel::__accumulate_binop_reduct< _BinOp >
- struct __gnu_parallel::__accumulate_selector< _It >
- struct __gnu_parallel::__adjacent_difference_selector< _It >
- struct __gnu_parallel::__count_if_selector< _It, _Diff >
- struct __gnu_parallel::__count_selector< _It, _Diff >
- struct __gnu_parallel::__fill_selector< _It >
- struct __gnu_parallel::__for_each_selector< _It >
- struct __gnu_parallel::__generate_selector< _It >
- struct __gnu_parallel::__generic_for_each_selector< _It >
- struct __gnu_parallel::__identity_selector< _It >
- struct __gnu_parallel::__inner_product_selector< _It, _It2, _Tp >
- struct __gnu_parallel::__max_element_reduct< _Compare, _It >
- struct __gnu_parallel::__min_element_reduct< _Compare, _It >
- struct __gnu_parallel::__replace_if_selector< _It, _Op, _Tp >
- struct __gnu_parallel::__replace_selector< _It, _Tp >
- struct __gnu_parallel::__transform1_selector< _It >
- struct __gnu_parallel::__transform2_selector< _It >
- struct __gnu_parallel::_DummyReduct
- struct __gnu_parallel::_Nothing

Namespaces

- namespace __gnu_parallel

6.615.1 Detailed Description

Functors representing different tasks to be plugged into the generic parallelization methods for embarrassingly parallel functions. This file is a GNU parallel extension to the Standard C++ Library.

6.616 iterator.h File Reference

Classes

- class __gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >
- class __gnu_parallel::_IteratorTriple< _Iterator1, _Iterator2, _Iterator3, _IteratorCategory >

Namespaces

- namespace __gnu_parallel

6.616.1 Detailed Description

Helper iterator classes for the std::transform() functions. This file is a GNU parallel extension to the Standard C++ Library.

6.617 list_partition.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _IIter>
  void __gnu_parallel::__shrink (std::vector< _IIter > &__os_starts, size_t &__count_to_two, size_t &__range_length)
- template<typename _IIter>
  void __gnu_parallel::__shrink_and_double (std::vector< _IIter > &__os_starts, size_t &__count_to_two, size_t &__range_length, const bool __make_twice)
- template<typename _IIter, typename _FunctorType>
  size_t __gnu_parallel::list_partition (const _IIter __begin, const _IIter __end, _IIter *__starts, size_t *__lengths, const int __num_parts, _FunctorType &__f, int __oversampling=0)

6.617.1 Detailed Description

_Functionality to split __sequence referenced by only input iterators. This file is a GNU parallel extension to the Standard C++ Library.

6.618 losertree.h File Reference

Classes

- struct __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser
- struct __gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >::_Loser
- class __gnu_parallel::_LoserTree< __stable, _Tp, _Compare >
- class __gnu_parallel::_LoserTree< false, _Tp, _Compare >
- class __gnu_parallel::_LoserTreeBase< _Tp, _Compare >
- class __gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >
- class __gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >
- class __gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >
- class __gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >
- class __gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >
- class __gnu_parallel::_LoserTreePointerUnguardedBase< _Tp, _Compare >
- class __gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >
- class __gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >
- class __gnu_parallel::_LoserTreeUnguardedBase< _Tp, _Compare >

Namespaces

- namespace __gnu_parallel

6.618.1 Detailed Description

Many generic loser tree variants. This file is a GNU parallel extension to the Standard C++ Library.

6.619 merge.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare>
  _OutputIterator __gnu_parallel::__merge_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)
- template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare>
  _OutputIterator __gnu_parallel::__merge_advance_movc (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &↩
  __begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)
- template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare>
  _OutputIterator __gnu_parallel::__merge_advance_usual (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &↩
  __begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)
- template<typename _RAIter1, typename _RAIter3, typename _Compare>
  _RAIter3 __gnu_parallel::__parallel_merge_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter1 &__↩
  begin2, _RAIter1 __end2, _RAIter3 __target, typename std::iterator_traits< _RAIter1 >::difference_type __↩
  max_length, _Compare __comp)
- template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _Compare>
  _RAIter3 __gnu_parallel::__parallel_merge_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__↩
  begin2, _RAIter2 __end2, _RAIter3 __target, typename std::iterator_traits< _RAIter1 >::difference_type __↩
  max_length, _Compare __comp)

6.619.1 Detailed Description

Parallel implementation of std::merge(). This file is a GNU parallel extension to the Standard C++ Library.

6.620 multiseq_selection.h File Reference

Classes

- class __gnu_parallel::_Lexicographic< _T1, _T2, _Compare >
- class __gnu_parallel::_LexicographicReverse< _T1, _T2, _Compare >

Namespaces

- namespace __gnu_parallel

Macros

- #define **__S**(__i)
- #define **__S**(__i)

Functions

- template<typename _RanSeqs, typename _RankType, typename _RankIterator, typename _Compare>
  void __gnu_parallel::multiseq_partition (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __↩
  rank, _RankIterator __begin_offsets, _Compare __comp=std::less< typename std::iterator_traits< typename std::iterator_traits< _RanSeqs >::value_type::first_type >::value_type >())
- template<typename _Tp, typename _RanSeqs, typename _RankType, typename _Compare>
  _Tp __gnu_parallel::multiseq_selection (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankType &__offset, _Compare __comp=std::less< _Tp >())

6.620.1 Detailed Description

Functions to find elements of a certain global __rank in multiple sorted sequences. Also serves for splitting such sequence sets.
The algorithm description can be found in

P. J. Varman, S. D. Scheufler, B. R. Iyer, and G. R. Ricard. Merging Multiple Lists on Hierarchical-Memory Multiprocessors. Journal of Parallel and Distributed Computing, 12(2):171-177, 1991.
This file is a GNU parallel extension to the Standard C++ Library.

6.621 multiway_merge.h File Reference

Classes

- struct __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Com
- struct __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >
- struct __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Com
- struct __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >
- struct __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable, _RAIterIterator, _RAIter3, _Difference
- struct __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _C
- class __gnu_parallel::_GuardedIterator< _RAIter, _Compare >
- struct __gnu_parallel::_LoserTreeTraits< _Tp >
- struct __gnu_parallel::_SamplingSorter< __stable, _RAIter, _StrictWeakOrdering >
- struct __gnu_parallel::_SamplingSorter< false, _RAIter, _StrictWeakOrdering >

Namespaces

- namespace __gnu_parallel

Macros

- #define **_GLIBCXX_PARALLEL_DECISION**(__a, __b, __c, __d)
- #define _GLIBCXX_PARALLEL_LENGTH(__s)
- #define **_GLIBCXX_PARALLEL_MERGE_3_CASE**(__a, __b, __c, __c0, __c1)
- #define **_GLIBCXX_PARALLEL_MERGE_4_CASE**(__a, __b, __c, __d, __c0, __c1, __c2)

Functions

- template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare>
  _OutputIterator __gnu_parallel::__merge_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)
- template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
  _RAIter3 __gnu_parallel::__sequential_multiway_merge (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **__gnu_parallel::multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_tag __tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **__gnu_parallel::multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sampling_tag __tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut __gnu_parallel::multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **__gnu_parallel::multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **__gnu_parallel::multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))

- template<template< typename _RAI, typename _Cp > class iterator, typename _RAIterIterator, typename _RAIter3, typename _↩ DifferenceTp, typename _Compare>
  _RAIter3 __gnu_parallel::multiway_merge_3_variant (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)
- template<template< typename _RAI, typename _Cp > class iterator, typename _RAIterIterator, typename _RAIter3, typename _↩ DifferenceTp, typename _Compare>
  _RAIter3 __gnu_parallel::multiway_merge_4_variant (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)
- template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType>
  void __gnu_parallel::multiway_merge_exact_splitting (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_↩ end, _DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _↩ DifferenceType, _DifferenceType > > *__pieces)
- template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
  _RAIter3 __gnu_parallel::multiway_merge_loser_tree (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_↩ end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)
- template<typename _UnguardedLoserTree, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
  _RAIter3 __gnu_parallel::multiway_merge_loser_tree_sentinel (_RAIterIterator __seqs_begin, _RAIterIterator ↩ __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIter↩ Iterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)
- template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
  _RAIter3 __gnu_parallel::multiway_merge_loser_tree_unguarded (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIter↩ Iterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)
- template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType>
  void __gnu_parallel::multiway_merge_sampling_splitting (_RAIterIterator __seqs_begin, _RAIterIterator _↩ _seqs_end, _DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _DifferenceType, _DifferenceType > > *__pieces)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **__gnu_parallel::multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_tag __tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut __gnu_parallel::multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator ↩ __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **__gnu_parallel::multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **__gnu_parallel::multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPair↩ Iterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __↩ tag=parallel_tag(0))
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **__gnu_parallel::multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling_tag __tag)
- template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Splitter, type- name _Compare>
  _RAIter3 __gnu_parallel::parallel_multiway_merge (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _Splitter __splitter, _DifferenceTp __length, _Compare __comp, _ThreadIndex __num_↩ threads)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **__gnu_parallel::stable_multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator ↩ __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_tag __tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **__gnu_parallel::stable_multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator ↩ __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag)

- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **__gnu_parallel::stable_multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **__gnu_parallel::stable_multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **__gnu_parallel::stable_multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling_tag __tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **__gnu_parallel::stable_multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_tag __tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **__gnu_parallel::stable_multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **__gnu_parallel::stable_multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **__gnu_parallel::stable_multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **__gnu_parallel::stable_multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling_tag __tag)

6.621.1 Detailed Description

Implementation of sequential and parallel multiway merge.
Explanations on the high-speed merging routines in the appendix of
P. Sanders. Fast priority queues for cached memory. ACM Journal of Experimental Algorithmics, 5, 2000.
This file is a GNU parallel extension to the Standard C++ Library.

6.621.2 Macro Definition Documentation

_GLIBCXX_PARALLEL_LENGTH

```
#define _GLIBCXX_PARALLEL_LENGTH(
            __s)
```

Length of a sequence described by a pair of iterators.
Referenced by __gnu_parallel::__sequential_multiway_merge(), __gnu_parallel::multiway_merge_exact_splitting(), __gnu_parallel::multiway_merge_loser_tree(), __gnu_parallel::multiway_merge_sampling_splitting(), and __gnu_parallel::parallel_multiwa

6.622 multiway_mergesort.h File Reference

Classes

- struct __gnu_parallel::_Piece< _DifferenceTp >
- struct __gnu_parallel::_PMWMSSortingData< _RAIter >
- struct __gnu_parallel::_SplitConsistently< __exact, _RAIter, _Compare, _SortingPlacesIterator >
- struct __gnu_parallel::_SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIterator >
- struct __gnu_parallel::_SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator >

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _RAIter, typename _DifferenceTp>
  void __gnu_parallel::__determine_samples (_PMWMSSortingData< _RAIter > *__sd, _DifferenceTp __num_samples)
- template<bool __stable, bool __exact, typename _RAIter, typename _Compare>
  void __gnu_parallel::parallel_sort_mwms (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)
- template<bool __stable, bool __exact, typename _RAIter, typename _Compare>
  void __gnu_parallel::parallel_sort_mwms_pu (_PMWMSSortingData< _RAIter > *__sd, _Compare &__comp)

6.622.1 Detailed Description

Parallel multiway merge sort. This file is a GNU parallel extension to the Standard C++ Library.

6.623 numericfwd.h File Reference

Namespaces

- namespace std
- namespace std::__parallel

Functions

- template<typename _IIter, typename _Tp, typename _BinaryOper, typename _Tag>
  _Tp **std::__parallel::__accumulate_switch** (_IIter, _IIter, _Tp, _BinaryOper, _Tag)
- template<typename _IIter, typename _Tp, typename _Tag>
  _Tp **std::__parallel::__accumulate_switch** (_IIter, _IIter, _Tp, _Tag)
- template<typename _RAIter, typename _Tp, typename _BinaryOper>
  _Tp **std::__parallel::__accumulate_switch** (_RAIter, _RAIter, _Tp, _BinaryOper, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism=__gnu_parallel::parallel_unbalanced)
- template<typename _IIter, typename _OIter, typename _BinaryOper, typename _Tag1, typename _Tag2>
  _OIter **std::__parallel::__adjacent_difference_switch** (_IIter, _IIter, _OIter, _BinaryOper, _Tag1, _Tag2)
- template<typename _IIter, typename _OIter, typename _BinaryOper>
  _OIter **std::__parallel::__adjacent_difference_switch** (_IIter, _IIter, _OIter, _BinaryOper, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism=__gnu_parallel::parallel_unbalanced)
- template<typename _IIter1, typename _IIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2, typename _Tag1, typename _Tag2>
  _Tp **std::__parallel::__inner_product_switch** (_IIter1, _IIter1, _IIter2, _Tp, _BinaryFunction1, _BinaryFunction2, _Tag1, _Tag2)
- template<typename _RAIter1, typename _RAIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>
  _Tp **std::__parallel::__inner_product_switch** (_RAIter1, _RAIter1, _RAIter2, _Tp, _BinaryFunction1, _BinaryFunction2, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism=__gnu_parallel::parallel_unbal
- template<typename _IIter, typename _OIter, typename _BinaryOper, typename _Tag1, typename _Tag2>
  _OIter **std::__parallel::__partial_sum_switch** (_IIter, _IIter, _OIter, _BinaryOper, _Tag1, _Tag2)
- template<typename _IIter, typename _OIter, typename _BinaryOper>
  _OIter **std::__parallel::__partial_sum_switch** (_IIter, _IIter, _OIter, _BinaryOper, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter, typename _Tp>
  _Tp **std::__parallel::accumulate** (_IIter, _IIter, _Tp)

- template<typename _IIter, typename _Tp>
  _Tp **std::__parallel::accumulate** (_IIter, _IIter, _Tp, __gnu_parallel::_Parallelism)
- template<typename _IIter, typename _Tp>
  _Tp **std::__parallel::accumulate** (_IIter, _IIter, _Tp, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _Tp, typename _BinaryOper>
  _Tp **std::__parallel::accumulate** (_IIter, _IIter, _Tp, _BinaryOper)
- template<typename _IIter, typename _Tp, typename _BinaryOper>
  _Tp **std::__parallel::accumulate** (_IIter, _IIter, _Tp, _BinaryOper, __gnu_parallel::_Parallelism)
- template<typename _IIter, typename _Tp, typename _BinaryOper>
  _Tp **std::__parallel::accumulate** (_IIter, _IIter, _Tp, _BinaryOper, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OIter>
  _OIter **std::__parallel::adjacent_difference** (_IIter, _IIter, _OIter)
- template<typename _IIter, typename _OIter>
  _OIter **std::__parallel::adjacent_difference** (_IIter, _IIter, _OIter, __gnu_parallel::_Parallelism)
- template<typename _IIter, typename _OIter>
  _OIter **std::__parallel::adjacent_difference** (_IIter, _IIter, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OIter, typename _BinaryOper>
  _OIter **std::__parallel::adjacent_difference** (_IIter, _IIter, _OIter, _BinaryOper)
- template<typename _IIter, typename _OIter, typename _BinaryOper>
  _OIter **std::__parallel::adjacent_difference** (_IIter, _IIter, _OIter, _BinaryOper, __gnu_parallel::_Parallelism)
- template<typename _IIter, typename _OIter, typename _BinaryOper>
  _OIter **std::__parallel::adjacent_difference** (_IIter, _IIter, _OIter, _BinaryOper, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _Tp>
  _Tp **std::__parallel::inner_product** (_IIter1, _IIter1, _IIter2, _Tp)
- template<typename _IIter1, typename _IIter2, typename _Tp>
  _Tp **std::__parallel::inner_product** (_IIter1, _IIter1, _IIter2, _Tp, __gnu_parallel::_Parallelism)
- template<typename _IIter1, typename _IIter2, typename _Tp>
  _Tp **std::__parallel::inner_product** (_IIter1, _IIter1, _IIter2, _Tp, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>
  _Tp **std::__parallel::inner_product** (_IIter1, _IIter1, _IIter2, _Tp, _BinaryFunction1, _BinaryFunction2)
- template<typename _IIter1, typename _IIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>
  _Tp **std::__parallel::inner_product** (_IIter1, _IIter1, _IIter2, _Tp, _BinaryFunction1, _BinaryFunction2, __gnu_parallel::_Parallelism)
- template<typename _IIter1, typename _IIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>
  _Tp **std::__parallel::inner_product** (_IIter1, _IIter1, _IIter2, _Tp, _BinaryFunction1, _BinaryFunction2, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OIter>
  _OIter **std::__parallel::partial_sum** (_IIter, _IIter, _OIter __result)
- template<typename _IIter, typename _OIter>
  _OIter **std::__parallel::partial_sum** (_IIter, _IIter, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OIter, typename _BinaryOper>
  _OIter **std::__parallel::partial_sum** (_IIter, _IIter, _OIter, _BinaryOper)
- template<typename _IIter, typename _OIter, typename _BinaryOper>
  _OIter **std::__parallel::partial_sum** (_IIter, _IIter, _OIter, _BinaryOper, __gnu_parallel::sequential_tag)

6.623.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

6.624 omp_loop.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>
  _Op __gnu_parallel::__for_each_template_random_access_omp_loop (_RAIter __begin, _RAIter __end, _Op ↩
  __o, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator_traits< _RAIter >↩
  ::difference_type __bound)

6.624.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of an OpenMP for loop. This file is a GNU parallel extension to the Standard C++ Library.

6.625 omp_loop_static.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>
  _Op __gnu_parallel::__for_each_template_random_access_omp_loop_static (_RAIter __begin, _RAIter __end, _Op __o, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator_traits< _RAIter >↩
  ::difference_type __bound)

6.625.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of an OpenMP for loop with static scheduling. This file is a GNU parallel extension to the Standard C++ Library.

6.626 par_loop.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>
  _Op __gnu_parallel::__for_each_template_random_access_ed (_RAIter __begin, _RAIter __end, _Op __o, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator_traits< _RAIter >::difference_type __bound)

6.626.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of equal splitting. This file is a GNU parallel extension to the Standard C++ Library.

6.627 parallel.h File Reference

6.627.1 Detailed Description

End-user include file. Provides advanced settings and tuning options. This file is a GNU parallel extension to the Standard C++ Library.

6.628 partial_sum.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator __gnu_parallel::__parallel_partial_sum (_IIter __begin, _IIter __end, _OutputIterator __result, ↩
  _BinaryOperation __bin_op)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator __gnu_parallel::__parallel_partial_sum_basecase (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __bin_op, typename std::iterator_traits< _IIter >::value_type __value)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator __gnu_parallel::__parallel_partial_sum_linear (_IIter __begin, _IIter __end, _OutputIterator __↩
  result, _BinaryOperation __bin_op, typename std::iterator_traits< _IIter >::difference_type __n)

6.628.1 Detailed Description

Parallel implementation of std::partial_sum(), i.e. prefix sums. This file is a GNU parallel extension to the Standard C++ Library.

6.629 partition.h File Reference

Namespaces

- namespace __gnu_parallel

Macros

- #define _GLIBCXX_VOLATILE

Functions

- template<typename _RAIter, typename _Compare>
  void __gnu_parallel::__parallel_nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __↩
  comp)
- template<typename _RAIter, typename _Compare>
  void __gnu_parallel::__parallel_partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __↩
  comp)
- template<typename _RAIter, typename _Predicate>
  std::iterator_traits< _RAIter >::difference_type __gnu_parallel::__parallel_partition (_RAIter __begin, _RAIter ↩
  __end, _Predicate __pred, _ThreadIndex __num_threads)

6.629.1 Detailed Description

Parallel implementation of std::partition(), std::nth_element(), and std::partial_sort(). This file is a GNU parallel extension to the Standard C++ Library.

6.629.2 Macro Definition Documentation

_GLIBCXX_VOLATILE

```
#define _GLIBCXX_VOLATILE
```

Decide whether to declare certain variables volatile.
Referenced by __gnu_parallel::__parallel_partition().

6.630 queue.h File Reference

Classes

- class __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >

Namespaces

- namespace __gnu_parallel

Macros

- #define _GLIBCXX_VOLATILE

6.630.1 Detailed Description

Lock-free double-ended queue. This file is a GNU parallel extension to the Standard C++ Library.

6.630.2 Macro Definition Documentation

_GLIBCXX_VOLATILE

```
#define _GLIBCXX_VOLATILE
```

Decide whether to declare certain variable volatile in this file.

6.631 quicksort.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _RAIter, typename _Compare>
  void __gnu_parallel::__parallel_sort_qs (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex _↩ _num_threads)
- template<typename _RAIter, typename _Compare>
  void __gnu_parallel::__parallel_sort_qs_conquer (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)
- template<typename _RAIter, typename _Compare>
  std::iterator_traits< _RAIter >::difference_type __gnu_parallel::__parallel_sort_qs_divide (_RAIter __begin, _↩ RAIter __end, _Compare __comp, typename std::iterator_traits< _RAIter >::difference_type __pivot_rank, type- name std::iterator_traits< _RAIter >::difference_type __num_samples, _ThreadIndex __num_threads)

6.631.1 Detailed Description

Implementation of a unbalanced parallel quicksort (in-place). This file is a GNU parallel extension to the Standard C++ Library.

6.632 random_number.h File Reference

Classes

- class __gnu_parallel::_RandomNumber

Namespaces

- namespace __gnu_parallel

6.632.1 Detailed Description

Random number generator based on the Mersenne twister. This file is a GNU parallel extension to the Standard C++ Library.

6.633 random_shuffle.h File Reference

Classes

- struct __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >
- struct __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >

Namespaces

- namespace __gnu_parallel

Typedefs

- typedef unsigned short __gnu_parallel::_BinIndex

Functions

- template<typename _RAIter, typename _RandomNumberGenerator>
  void __gnu_parallel::__parallel_random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator __rng=_RandomNumber())
- template<typename _RAIter, typename _RandomNumberGenerator>
  void __gnu_parallel::__parallel_random_shuffle_drs (_RAIter __begin, _RAIter __end, typename std::iterator_traits< _RAIter >::difference_type __n, _ThreadIndex __num_threads, _RandomNumberGenerator &__rng)
- template<typename _RAIter, typename _RandomNumberGenerator>
  void __gnu_parallel::__parallel_random_shuffle_drs_pu (_DRSSorterPU< _RAIter, _RandomNumberGenerator > *__pus)
- template<typename _RandomNumberGenerator>
  int __gnu_parallel::__random_number_pow2 (int __logp, _RandomNumberGenerator &__rng)
- template<typename _Tp>
  _Tp __gnu_parallel::__round_up_to_pow2 (_Tp __x)
- template<typename _RAIter, typename _RandomNumberGenerator>
  void __gnu_parallel::__sequential_random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &__rng)

6.633.1 Detailed Description

Parallel implementation of std::random_shuffle(). This file is a GNU parallel extension to the Standard C++ Library.

6.634 search.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _RAIter, typename _DifferenceTp>
  void __gnu_parallel::__calc_borders (_RAIter __elements, _DifferenceTp __length, _DifferenceTp *__off)
- template<typename __RAIter1, typename __RAIter2, typename _Pred>
  __RAIter1 __gnu_parallel::__search_template (__RAIter1 __begin1, __RAIter1 __end1, __RAIter2 __begin2, ↩
  __RAIter2 __end2, _Pred __pred)

6.634.1 Detailed Description

Parallel implementation base for std::search() and std::search_n(). This file is a GNU parallel extension to the Standard C++ Library.

6.635 set_operations.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _IIter, typename _OutputIterator>
  _OutputIterator **__gnu_parallel::__copy_tail** (std::pair< _IIter, _IIter > __b, std::pair< _IIter, _IIter > __e, _↩
  OutputIterator __r)
- template<typename _IIter, typename _OutputIterator, typename _Compare>
  _OutputIterator **__gnu_parallel::__parallel_set_difference** (_IIter __begin1, _IIter __end1, _IIter __begin2, _↩
  IIter __end2, _OutputIterator __result, _Compare __comp)
- template<typename _IIter, typename _OutputIterator, typename _Compare>
  _OutputIterator **__gnu_parallel::__parallel_set_intersection** (_IIter __begin1, _IIter __end1, _IIter __begin2, _IIter __end2, _OutputIterator __result, _Compare __comp)
- template<typename _IIter, typename _OutputIterator, typename _Operation>
  _OutputIterator **__gnu_parallel::__parallel_set_operation** (_IIter __begin1, _IIter __end1, _IIter __begin2, _↩
  IIter __end2, _OutputIterator __result, _Operation __op)
- template<typename _IIter, typename _OutputIterator, typename _Compare>
  _OutputIterator **__gnu_parallel::__parallel_set_symmetric_difference** (_IIter __begin1, _IIter __end1, _IIter __begin2, _IIter __end2, _OutputIterator __result, _Compare __comp)
- template<typename _IIter, typename _OutputIterator, typename _Compare>
  _OutputIterator **__gnu_parallel::__parallel_set_union** (_IIter __begin1, _IIter __end1, _IIter __begin2, _IIter __end2, _OutputIterator __result, _Compare __comp)

6.635.1 Detailed Description

Parallel implementations of set operations for random-access iterators. This file is a GNU parallel extension to the Standard C++ Library.

6.636 settings.h File Reference

Classes

- struct __gnu_parallel::_Settings

Namespaces

- namespace __gnu_parallel

Macros

- #define _GLIBCXX_PARALLEL_CONDITION(__c)

6.636.1 Detailed Description

Runtime settings and tuning parameters, heuristics to decide whether to use parallelized algorithms.
This file is a GNU parallel extension to the Standard C++ Library.

6.636.2 Deciding whether to run an algorithm in parallel.

There are several ways the user can switch on and off the parallel execution of an algorithm, both at compile- and run-time.
Only sequential execution can be forced at compile-time. This reduces code size and protects code parts that have non-thread-safe side effects.
Ultimately, forcing parallel execution at compile-time makes sense. Often, the sequential algorithm implementation is used as a subroutine, so no reduction in code size can be achieved. Also, the machine the program is run on might have only one processor core, so to avoid overhead, the algorithm is executed sequentially.
To force sequential execution of an algorithm ultimately at compile-time, the user must add the tag gnu_parallel↩
::sequential_tag() to the end of the parameter list, e. g.

```
std::sort(__v.begin(), __v.end(), __gnu_parallel::sequential_tag());
```

This is compatible with all overloaded algorithm variants. No additional code will be instantiated, at all. The same holds for most algorithm calls with iterators not providing random access.
If the algorithm call is not forced to be executed sequentially at compile-time, the decision is made at run-time. The global variable __gnu_parallel::_Settings::algorithm_strategy is checked. It is a tristate variable corresponding to:

- a. force_sequential, meaning the sequential algorithm is executed.
- b. force_parallel, meaning the parallel algorithm is executed.
- c. heuristic

For heuristic, the parallel algorithm implementation is called only if the input size is sufficiently large. For most algorithms, the input size is the (combined) length of the input sequence(__s). The threshold can be set by the user, individually for each algorithm. The according variables are called gnu_parallel::_Settings::[algorithm]_minimal_n .
For some of the algorithms, there are even more tuning options, e. g. the ability to choose from multiple algorithm variants. See below for details.

6.636.3 Macro Definition Documentation

_GLIBCXX_PARALLEL_CONDITION

```
#define _GLIBCXX_PARALLEL_CONDITION(
              __c)
```

Determine at compile(?)-time if the parallel variant of an algorithm should be called.

Parameters

_↩ _c	A condition that is convertible to bool that is overruled by __gnu_parallel::_Settings::algorithm_strategy. Usually a decision based on the input size.

6.637 sort.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<bool __stable, typename _RAIter, typename _Compare, typename _Parallelism>
  void **__gnu_parallel::__parallel_sort** (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __↩
  parallelism)
- template<bool __stable, typename _RAIter, typename _Compare>
  void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, balanced_quicksort_tag
  __parallelism)
- template<bool __stable, typename _RAIter, typename _Compare>
  void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, default_parallel_tag
  __parallelism)
- template<bool __stable, typename _RAIter, typename _Compare>
  void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_exact_tag
  __parallelism)
- template<bool __stable, typename _RAIter, typename _Compare>
  void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_sampling_tag
  __parallelism)
- template<bool __stable, typename _RAIter, typename _Compare>
  void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_tag
  __parallelism)
- template<bool __stable, typename _RAIter, typename _Compare>
  void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, parallel_tag __↩
  parallelism)
- template<bool __stable, typename _RAIter, typename _Compare>
  void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, quicksort_tag __↩
  parallelism)

6.637.1 Detailed Description

Parallel sorting algorithm switch. This file is a GNU parallel extension to the Standard C++ Library.

6.638 tags.h File Reference

Classes

- struct __gnu_parallel::balanced_quicksort_tag
- struct __gnu_parallel::balanced_tag
- struct __gnu_parallel::constant_size_blocks_tag
- struct __gnu_parallel::default_parallel_tag
- struct __gnu_parallel::equal_split_tag
- struct __gnu_parallel::exact_tag
- struct __gnu_parallel::find_tag
- struct __gnu_parallel::growing_blocks_tag
- struct __gnu_parallel::multiway_mergesort_exact_tag
- struct __gnu_parallel::multiway_mergesort_sampling_tag
- struct __gnu_parallel::multiway_mergesort_tag
- struct __gnu_parallel::omp_loop_static_tag
- struct __gnu_parallel::omp_loop_tag
- struct __gnu_parallel::parallel_tag
- struct __gnu_parallel::quicksort_tag
- struct __gnu_parallel::sampling_tag
- struct __gnu_parallel::sequential_tag
- struct __gnu_parallel::unbalanced_tag

Namespaces

- namespace __gnu_parallel

6.638.1 Detailed Description

Tags for compile-time selection. This file is a GNU parallel extension to the Standard C++ Library.

6.639 types.h File Reference

Namespaces

- namespace __gnu_parallel

Typedefs

- typedef int64_t __gnu_parallel::_CASable
- typedef uint64_t __gnu_parallel::_SequenceIndex
- typedef uint16_t __gnu_parallel::_ThreadIndex

Enumerations

- enum __gnu_parallel::_AlgorithmStrategy { **heuristic** , **force_sequential** , **force_parallel** }
- enum __gnu_parallel::_FindAlgorithm { **GROWING_BLOCKS** , **CONSTANT_SIZE_BLOCKS** , **EQUAL_SPLIT** }
- enum __gnu_parallel::_MultiwayMergeAlgorithm { **LOSER_TREE** }
- enum __gnu_parallel::_Parallelism { __gnu_parallel::sequential , __gnu_parallel::parallel_unbalanced , __gnu_parallel::parallel_balanced , __gnu_parallel::parallel_omp_loop , __gnu_parallel::parallel_omp_loop_static , __gnu_parallel::parallel_taskqueue }
- enum __gnu_parallel::_PartialSumAlgorithm { **RECURSIVE** , **LINEAR** }
- enum __gnu_parallel::_SortAlgorithm { **MWMS** , **QS** , **QS_BALANCED** }
- enum __gnu_parallel::_SplittingAlgorithm { **SAMPLING** , **EXACT** }

Variables

- static const int __gnu_parallel::_CASable_bits
- static const _CASable __gnu_parallel::_CASable_mask

6.639.1 Detailed Description

Basic types and typedefs. This file is a GNU parallel extension to the Standard C++ Library.

6.640 unique_copy.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _IIter, class _OutputIterator> _OutputIterator __gnu_parallel::__parallel_unique_copy (_IIter __first, _IIter __last, _OutputIterator __result)
- template<typename _IIter, class _OutputIterator, class _BinaryPredicate> _OutputIterator __gnu_parallel::__parallel_unique_copy (_IIter __first, _IIter __last, _OutputIterator __result, _↩ BinaryPredicate __binary_pred)

6.640.1 Detailed Description

Parallel implementations of std::unique_copy(). This file is a GNU parallel extension to the Standard C++ Library.

6.641 workstealing.h File Reference

Classes

- struct __gnu_parallel::_Job< _DifferenceTp >

Namespaces

- namespace __gnu_parallel

Macros

- #define **_GLIBCXX_JOB_VOLATILE**

Functions

- template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>
  _Op __gnu_parallel::__for_each_template_random_access_workstealing (_RAIter __begin, _RAIter __end, _Op __op, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator_traits< _RAIter >::difference_type __bound)

6.641.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of work-stealing.
Work stealing is described in
R. D. Blumofe and C. E. Leiserson. Scheduling multithreaded computations by work stealing. Journal of the ACM, 46(5):720-748, 1999.
This file is a GNU parallel extension to the Standard C++ Library.

6.642 print File Reference

Macros

- #define **__glibcxx_want_print**
- #define **_GLIBCXX_PRINT**

6.642.1 Detailed Description

This is a Standard C++ Library header.

6.643 queue File Reference

Macros

- #define **__glibcxx_want_adaptor_iterator_pair_constructor**
- #define **__glibcxx_want_containers_ranges**
- #define **_GLIBCXX_QUEUE**

6.643.1 Detailed Description

This is a Standard C++ Library header.

6.644 experimental/random File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **__cpp_lib_experimental_randint**
- #define **_GLIBCXX_EXPERIMENTAL_RANDOM**

Functions

- std::default_random_engine & **std::experimental::_S_randint_engine** ()
- template<typename _IntType>
  _IntType **std::experimental::randint** (_IntType __a, _IntType __b)
- void **std::experimental::reseed** ()
- void **std::experimental::reseed** (default_random_engine::result_type __value)

6.644.1 Detailed Description

This is a TS C++ Library header.

6.645 random File Reference

Macros

- #define **__glibcxx_want_philox_engine**
- #define **_GLIBCXX_RANDOM**

6.645.1 Detailed Description

This is a Standard C++ Library header.

6.646 ranges File Reference

Macros

- #define **_GLIBCXX_RANGES**

6.646.1 Detailed Description

This is a Standard C++ Library header.

6.647 experimental/ratio File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **_GLIBCXX_EXPERIMENTAL_RATIO**

Variables

- template<typename _R1, typename _R2>
  constexpr bool **std::experimental::ratio_equal_v**
- template<typename _R1, typename _R2>
  constexpr bool **std::experimental::ratio_greater_equal_v**
- template<typename _R1, typename _R2>
  constexpr bool **std::experimental::ratio_greater_v**
- template<typename _R1, typename _R2>
  constexpr bool **std::experimental::ratio_less_equal_v**
- template<typename _R1, typename _R2>
  constexpr bool **std::experimental::ratio_less_v**
- template<typename _R1, typename _R2>
  constexpr bool **std::experimental::ratio_not_equal_v**

6.647.1 Detailed Description

This is a TS C++ Library header.

6.648 ratio File Reference

Classes

- struct std::ratio< _Num, _Den >
- struct std::ratio_equal< _R1, _R2 >
- struct std::ratio_greater< _R1, _R2 >
- struct std::ratio_greater_equal< _R1, _R2 >
- struct std::ratio_less< _R1, _R2 >
- struct std::ratio_less_equal< _R1, _R2 >
- struct std::ratio_not_equal< _R1, _R2 >

Namespaces

- namespace std

Macros

- #define **__glibcxx_want_ratio**
- #define **_GLIBCXX_RATIO**

Typedefs

- using **std::atto**
- using **std::centi**
- using **std::deca**
- using **std::deci**
- using **std::exa**
- using **std::femto**
- using **std::giga**
- using **std::hecto**
- using **std::kilo**
- using **std::mega**
- using **std::micro**
- using **std::milli**

- using **std::nano**
- using **std::peta**
- using **std::pico**
- template<typename _R1, typename _R2>
  using std::ratio_add
- template<typename _R1, typename _R2>
  using std::ratio_divide
- template<typename _R1, typename _R2>
  using std::ratio_multiply
- template<typename _R1, typename _R2>
  using std::ratio_subtract
- using **std::tera**

Variables

- template<typename _R1, typename _R2>
  constexpr bool **std::ratio_equal_v**
- template<typename _R1, typename _R2>
  constexpr bool **std::ratio_greater_equal_v**
- template<typename _R1, typename _R2>
  constexpr bool **std::ratio_greater_v**
- template<typename _R1, typename _R2>
  constexpr bool **std::ratio_less_equal_v**
- template<typename _R1, typename _R2>
  constexpr bool **std::ratio_less_v**
- template<typename _R1, typename _R2>
  constexpr bool **std::ratio_not_equal_v**

6.648.1 Detailed Description

This is a Standard C++ Library header.

6.649 tr2/ratio File Reference

Namespaces

- namespace std
- namespace std::tr2

6.649.1 Detailed Description

This is a TR2 C++ Library header.

6.650 experimental/regex File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **_GLIBCXX_EXPERIMENTAL_REGEX**

Typedefs

- typedef match_results< const char * > **std::experimental::fundamentals_v2::pmr::cmatch**
- template<typename _BidirectionalIterator>
  using **std::experimental::fundamentals_v2::pmr::match_results**
- typedef match_results< string::const_iterator > **std::experimental::fundamentals_v2::pmr::smatch**
- typedef match_results< const wchar_t * > **std::experimental::fundamentals_v2::pmr::wcmatch**
- typedef match_results< wstring::const_iterator > **std::experimental::fundamentals_v2::pmr::wsmatch**

6.650.1 Detailed Description

This is a TS C++ Library header.

6.651 regex File Reference

Namespaces

- namespace std

Macros

- #define **__glibcxx_want_nonmember_container_access**
- #define **_GLIBCXX_REGEX**

Typedefs

- using **std::pmr::cmatch**
- template<typename _BidirectionalIterator>
  using **std::pmr::match_results**
- using **std::pmr::smatch**
- using **std::pmr::wcmatch**
- using **std::pmr::wsmatch**

6.651.1 Detailed Description

This is a Standard C++ Library header.

6.652 scoped_allocator File Reference

Classes

- class std::scoped_allocator_adaptor< _OuterAlloc, _InnerAllocs >

Namespaces

- namespace std

Macros

- #define **__glibcxx_want_allocator_traits_is_always_equal**
- #define **_SCOPED_ALLOCATOR**

6.652.1 Detailed Description

This is a Standard C++ Library header.

6.653 semaphore File Reference

Macros

- #define **__glibcxx_want_semaphore**
- #define **_GLIBCXX_SEMAPHORE**

6.653.1 Detailed Description

This is a Standard C++ Library header.

6.654 debug/set File Reference

Namespaces

- namespace std
- namespace std::__debug

Macros

- #define **_GLIBCXX_DEBUG_SET**

6.654.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

6.655 experimental/set File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **_GLIBCXX_EXPERIMENTAL_SET**

Typedefs

- template<typename _Key, typename _Compare = less<_Key>>
  using **std::experimental::fundamentals_v2::pmr::multiset**
- template<typename _Key, typename _Compare = less<_Key>>
  using **std::experimental::fundamentals_v2::pmr::set**

Functions

- template<typename _Key, typename _Compare, typename _Alloc, typename _Predicate>
  void **std::experimental::erase_if** (multiset< _Key, _Compare, _Alloc > &__cont, _Predicate __pred)
- template<typename _Key, typename _Compare, typename _Alloc, typename _Predicate>
  void **std::experimental::erase_if** (set< _Key, _Compare, _Alloc > &__cont, _Predicate __pred)

6.655.1 Detailed Description

This is a TS C++ Library header.

6.656 set File Reference

Namespaces

- namespace std

Macros

- #define **__glibcxx_want_allocator_traits_is_always_equal**
- #define **__glibcxx_want_containers_ranges**
- #define **__glibcxx_want_erase_if**
- #define **__glibcxx_want_generic_associative_lookup**
- #define **__glibcxx_want_node_extract**
- #define **__glibcxx_want_nonmember_container_access**
- #define **_GLIBCXX_SET**

Typedefs

- template<typename _Key, typename _Cmp = std::less<_Key>>
  using **std::pmr::multiset**
- template<typename _Key, typename _Cmp = std::less<_Key>>
  using **std::pmr::set**

Functions

- template<typename _Key, typename _Compare, typename _Alloc, typename _Predicate>
  multiset< _Key, _Compare, _Alloc >::size_type **std::erase_if** (multiset< _Key, _Compare, _Alloc > &__cont, _Predicate __pred)
- template<typename _Key, typename _Compare, typename _Alloc, typename _Predicate>
  set< _Key, _Compare, _Alloc >::size_type **std::erase_if** (set< _Key, _Compare, _Alloc > &__cont, _Predicate __pred)

6.656.1 Detailed Description

This is a Standard C++ Library header.

6.657 shared_mutex File Reference

Classes

- class std::shared_lock< _Mutex >
- class std::shared_timed_mutex

Namespaces

- namespace std

Macros

- #define **__glibcxx_want_shared_mutex**
- #define **__glibcxx_want_shared_timed_mutex**
- #define **_GLIBCXX_SHARED_MUTEX**

6.657.1 Detailed Description

This is a Standard C++ Library header.

6.658 source_location File Reference

Macros

- #define **__glibcxx_want_source_location**
- #define **_GLIBCXX_SRCLOC**

6.658.1 Detailed Description

This is a Standard C++ Library header.

6.659 span File Reference

Macros

- #define **__glibcxx_want_span**
- #define **__glibcxx_want_span_initializer_list**
- #define **_GLIBCXX_SPAN**

6.659.1 Detailed Description

This is a Standard C++ Library header.

6.660 sstream File Reference

Classes

- class std::basic_istringstream< _CharT, _Traits, _Alloc >
- class std::basic_ostringstream< _CharT, _Traits, _Alloc >
- class std::basic_stringbuf< _CharT, _Traits, _Alloc >
- class std::basic_stringstream< _CharT, _Traits, _Alloc >

Namespaces

- namespace std

Macros

- #define **__glibcxx_want_sstream_from_string_view**
- #define **_GLIBCXX_LVAL_REF_QUAL**
- #define **_GLIBCXX_SSTREAM**
- #define **_GLIBCXX_SSTREAM_ALWAYS_INLINE**

Functions

- template<class _CharT, class _Traits, class _Allocator>
  void std::swap (basic_istringstream< _CharT, _Traits, _Allocator > &__x, basic_istringstream< _CharT, _Traits, _Allocator > &__y)
- template<class _CharT, class _Traits, class _Allocator>
  void std::swap (basic_ostringstream< _CharT, _Traits, _Allocator > &__x, basic_ostringstream< _CharT, _Traits, _Allocator > &__y)
- template<class _CharT, class _Traits, class _Allocator>
  void std::swap (basic_stringbuf< _CharT, _Traits, _Allocator > &__x, basic_stringbuf< _CharT, _Traits, _Allocator > &__y) noexcept(noexcept(__x.swap(__y)))
- template<class _CharT, class _Traits, class _Allocator>
  void std::swap (basic_stringstream< _CharT, _Traits, _Allocator > &__x, basic_stringstream< _CharT, _Traits, _Allocator > &__y)

6.660.1 Detailed Description

This is a Standard C++ Library header.

6.661 stack File Reference

Macros

- #define **__glibcxx_want_adaptor_iterator_pair_constructor**
- #define **__glibcxx_want_containers_ranges**
- #define **_GLIBCXX_STACK**

6.661.1 Detailed Description

This is a Standard C++ Library header.

6.662 stdatomic.h File Reference

6.662.1 Detailed Description

This is a Standard C++ Library header.

6.663 stdbit.h File Reference

6.663.1 Detailed Description

This is a Standard C++ Library header.

6.664 stdckdint.h File Reference

6.664.1 Detailed Description

This is a Standard C++ Library header.

6.665 stdexcept File Reference

Classes

- class std::domain_error
- class std::invalid_argument
- class std::length_error
- class std::logic_error
- class std::out_of_range
- class std::overflow_error
- class std::range_error
- class std::runtime_error
- class std::underflow_error

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_STDEXCEPT**

Typedefs

- typedef basic_string< char > **std::__sso_string**

6.665.1 Detailed Description

This is a Standard C++ Library header.

6.666 stdlib.h File Reference

Functions

- template<typename _Tp>
  _Tp abs (const complex< _Tp > &)
- ldiv_t **div** (long __i, long __j) noexcept

6.666.1 Detailed Description

This is a Standard C++ Library header.

6.666.2 Function Documentation

abs()

```
template<typename _Tp>
_Tp std::abs (
            const complex< _Tp > & __z)  [inline]
```

Return magnitude of *z*.

6.667 stop_token File Reference

6.667.1 Detailed Description

This is a Standard C++ Library header.

6.668 streambuf File Reference

Classes

- class std::basic_streambuf< _CharT, _Traits >

Namespaces

- namespace std

Macros

- #define **_GLIBXX_STREAMBUF**
- #define **_IsUnused**

Functions

- template<typename _CharT, typename _Traits>
  streamsize **std::__copy_streambufs_eof** (basic_streambuf< _CharT, _Traits > *__sbin, basic_streambuf< ↩ _CharT, _Traits > *__sbout, bool &__ineof)
- template<> streamsize **std::__copy_streambufs_eof** (basic_streambuf< char > *__sbin, basic_streambuf< char > *__sbout, bool &__ineof)

- template<> streamsize **std::__copy_streambufs_eof** (basic_streambuf< wchar_t > *__sbin, basic_streambuf< wchar_t > *__sbout, bool &__ineof)

6.668.1 Detailed Description

This is a Standard C++ Library header.

6.669 debug/string File Reference

Classes

- class __gnu_debug::basic_string< _CharT, _Traits, _Allocator >
- struct std::hash< __gnu_debug::basic_string< _CharT > >

Namespaces

- namespace __gnu_debug
- namespace std

Macros

- #define **__glibcxx_check_string**(_String)
- #define **__glibcxx_check_string_constructor**(_Str)
- #define **__glibcxx_check_string_len**(_String, _Len)
- #define **__glibcxx_check_string_n_constructor**(_Str, _Size)
- #define **_GLIBCXX_DEBUG_STRING**
- #define **_GLIBCXX_DEBUG_VERIFY_STR_COND_AT**(_Cond, _File, _Line, _Func)
- #define **_GLIBCXX_INSERT_RETURNS_ITERATOR**
- #define **_GLIBCXX_INSERT_RETURNS_ITERATOR_ONLY**(expr)

Typedefs

- typedef basic_string< char > **__gnu_debug::string**
- typedef basic_string< char16_t > __gnu_debug::u16string
- typedef basic_string< char32_t > __gnu_debug::u32string
- typedef basic_string< wchar_t > **__gnu_debug::wstring**

Functions

- template<typename _CharT, typename _Integer>
  const _CharT * __gnu_debug::__check_string (const _CharT *__s, _Integer __n, const char *__file, unsigned int __line, const char *__function)
- template<typename _CharT>
  const _CharT * __gnu_debug::__check_string (const _CharT *__s, const char *__file, unsigned int __line, const char *__function)
- template<typename _CharT, typename _Traits, typename _Allocator>
  std::basic_istream< _CharT, _Traits > & **__gnu_debug::getline** (std::basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Allocator > &__str)
- template<typename _CharT, typename _Traits, typename _Allocator>
  std::basic_istream< _CharT, _Traits > & **__gnu_debug::getline** (std::basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Allocator > &__str, _CharT __delim)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **__gnu_debug::operator!=** (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__↩
  rhs)

- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **__gnu_debug::operator!=** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **__gnu_debug::operator!=** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  basic_string< _CharT, _Traits, _Allocator > **__gnu_debug::operator+** (_CharT __lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  basic_string< _CharT, _Traits, _Allocator > **__gnu_debug::operator+** (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  basic_string< _CharT, _Traits, _Allocator > **__gnu_debug::operator+** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, _CharT __rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  basic_string< _CharT, _Traits, _Allocator > **__gnu_debug::operator+** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  basic_string< _CharT, _Traits, _Allocator > **__gnu_debug::operator+** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **__gnu_debug::operator<** (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **__gnu_debug::operator<** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **__gnu_debug::operator<** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  std::basic_ostream< _CharT, _Traits > & **__gnu_debug::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const basic_string< _CharT, _Traits, _Allocator > &__str)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **__gnu_debug::operator<=** (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **__gnu_debug::operator<=** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **__gnu_debug::operator<=** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **__gnu_debug::operator==** (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **__gnu_debug::operator==** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **__gnu_debug::operator==** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **__gnu_debug::operator>** (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)

- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **__gnu_debug::operator>** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__↩
  rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **__gnu_debug::operator>** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string<
  _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **__gnu_debug::operator>=** (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &_↩
  _rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **__gnu_debug::operator>=** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *_↩
  _rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **__gnu_debug::operator>=** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string<
  _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  std::basic_istream< _CharT, _Traits > & **__gnu_debug::operator>>** (std::basic_istream< _CharT, _Traits >
  &__is, basic_string< _CharT, _Traits, _Allocator > &__str)
- template<typename _CharT, typename _Traits, typename _Allocator>
  void **__gnu_debug::swap** (basic_string< _CharT, _Traits, _Allocator > &__lhs, basic_string< _CharT, _Traits,
  _Allocator > &__rhs)

6.669.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

6.670 experimental/string File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **_GLIBCXX_EXPERIMENTAL_STRING**

Typedefs

- template<typename _CharT, typename _Traits = char_traits<_CharT>>
  using **std::experimental::fundamentals_v2::pmr::basic_string**
- typedef basic_string< char > **std::experimental::fundamentals_v2::pmr::string**
- typedef basic_string< char16_t > **std::experimental::fundamentals_v2::pmr::u16string**
- typedef basic_string< char32_t > **std::experimental::fundamentals_v2::pmr::u32string**
- typedef basic_string< wchar_t > **std::experimental::fundamentals_v2::pmr::wstring**

Functions

- template<typename _CharT, typename _Traits, typename _Alloc, typename _Up>
  void **std::experimental::erase** (basic_string< _CharT, _Traits, _Alloc > &__cont, const _Up &__value)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _Predicate>
  void **std::experimental::erase_if** (basic_string< _CharT, _Traits, _Alloc > &__cont, _Predicate __pred)

6.670.1 Detailed Description

This is a TS C++ Library header.

6.671 string File Reference

Namespaces

- namespace std

Macros

- #define **__glibcxx_want_algorithm_default_value_type**
- #define **__glibcxx_want_allocator_traits_is_always_equal**
- #define **__glibcxx_want_constexpr_char_traits**
- #define **__glibcxx_want_constexpr_string**
- #define **__glibcxx_want_containers_ranges**
- #define **__glibcxx_want_erase_if**
- #define **__glibcxx_want_nonmember_container_access**
- #define **__glibcxx_want_string_resize_and_overwrite**
- #define **__glibcxx_want_string_udls**
- #define **__glibcxx_want_to_string**
- #define **_GLIBCXX_STRING**

Typedefs

- template<typename _CharT, typename _Traits = char_traits<_CharT>>
  using **std::pmr::basic_string**
- using **std::pmr::string**
- using **std::pmr::u16string**
- using **std::pmr::u32string**
- using **std::pmr::wstring**

6.671.1 Detailed Description

This is a Standard C++ Library header.

6.672 experimental/string_view File Reference

Classes

- class std::experimental::fundamentals_v1::basic_string_view< _CharT, _Traits >

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **__cpp_lib_experimental_string_view**
- #define **_GLIBCXX_EXPERIMENTAL_STRING_VIEW**

Typedefs

- using **std::experimental::string_view**
- using **std::experimental::u16string_view**
- using **std::experimental::u32string_view**
- using **std::experimental::wstring_view**

Functions

- template<typename _CharT, typename _Traits>
  constexpr bool **std::experimental::operator!=** (__type_identity_t< basic_string_view< _CharT, _Traits > > ↩
  __x, basic_string_view< _CharT, _Traits > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **std::experimental::operator!=** (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **std::experimental::operator!=** (basic_string_view< _CharT, _Traits > __x, basic_string_view< _CharT, _Traits > __y) noexcept
- constexpr basic_string_view< char > **std::experimental::literals::operator""sv** (const char *__str, size_t __↩
  len) noexcept
- constexpr basic_string_view< char16_t > **std::experimental::literals::operator""sv** (const char16_t *__str, size_t __len) noexcept
- constexpr basic_string_view< char32_t > **std::experimental::literals::operator""sv** (const char32_t *__str, size_t __len) noexcept
- constexpr basic_string_view< wchar_t > **std::experimental::literals::operator""sv** (const wchar_t *__str, size_t __len) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **std::experimental::operator<** (__type_identity_t< basic_string_view< _CharT, _Traits > > _↩
  _x, basic_string_view< _CharT, _Traits > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **std::experimental::operator<** (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **std::experimental::operator<** (basic_string_view< _CharT, _Traits > __x, basic_string_view< _CharT, _Traits > __y) noexcept
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::experimental::operator<<** (basic_ostream< _CharT, _Traits > &↩
  __os, basic_string_view< _CharT, _Traits > __str)
- template<typename _CharT, typename _Traits>
  constexpr bool **std::experimental::operator<=** (__type_identity_t< basic_string_view< _CharT, _Traits > > ↩
  __x, basic_string_view< _CharT, _Traits > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **std::experimental::operator<=** (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **std::experimental::operator<=** (basic_string_view< _CharT, _Traits > __x, basic_string_view< _CharT, _Traits > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **std::experimental::operator==** (__type_identity_t< basic_string_view< _CharT, _Traits > > ↩
  __x, basic_string_view< _CharT, _Traits > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **std::experimental::operator==** (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept

- template<typename _CharT, typename _Traits>
  constexpr bool **std::experimental::operator==** (basic_string_view< _CharT, _Traits > __x, basic_string_view< _CharT, _Traits > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **std::experimental::operator>** (__type_identity_t< basic_string_view< _CharT, _Traits > > _↩ _x, basic_string_view< _CharT, _Traits > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **std::experimental::operator>** (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **std::experimental::operator>** (basic_string_view< _CharT, _Traits > __x, basic_string_view< _CharT, _Traits > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **std::experimental::operator>=** (__type_identity_t< basic_string_view< _CharT, _Traits > > __x, basic_string_view< _CharT, _Traits > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **std::experimental::operator>=** (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **std::experimental::operator>=** (basic_string_view< _CharT, _Traits > __x, basic_string_view< _CharT, _Traits > __y) noexcept

6.672.1 Detailed Description

This is a TS C++ Library header.

6.673 string_view File Reference

Classes

- class std::basic_string_view< _CharT, _Traits >

Namespaces

- namespace std
- namespace std::literals

Macros

- #define **__glibcxx_want_constexpr_char_traits**
- #define **__glibcxx_want_constexpr_string_view**
- #define **__glibcxx_want_freestanding_string_view**
- #define **__glibcxx_want_starts_ends_with**
- #define **__glibcxx_want_string_contains**
- #define **__glibcxx_want_string_view**
- #define **_GLIBCXX_STRING_VIEW**

Typedefs

- using **std::string_view**
- using **std::u16string_view**
- using **std::u32string_view**
- using **std::wstring_view**

Functions

- constexpr size_t **std::__sv_check** (size_t __size, size_t __pos, const char *__s)
- constexpr size_t **std::__sv_limit** (size_t __size, size_t __pos, size_t __off) noexcept
- constexpr basic_string_view< char > **std::literals::operator""sv** (const char *__str, size_t __len) noexcept
- constexpr basic_string_view< char16_t > **std::literals::operator""sv** (const char16_t *__str, size_t __len) noexcept
- constexpr basic_string_view< char32_t > **std::literals::operator""sv** (const char32_t *__str, size_t __len) noexcept
- constexpr basic_string_view< wchar_t > **std::literals::operator""sv** (const wchar_t *__str, size_t __len) noexcept
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::operator<<** (basic_ostream< _CharT, _Traits > &__os, basic_string_view< _CharT, _Traits > __str)
- template<typename _CharT, typename _Traits>
  constexpr auto **std::operator<=>** (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept -> decltype(__detail::__char_traits_cmp_cat< _Traits >(0))
- template<typename _CharT, typename _Traits>
  constexpr bool **std::operator==** (basic_string_view< _CharT, _Traits > __x, type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept

6.673.1 Detailed Description

This is a Standard C++ Library header.

6.674 syncstream File Reference

Macros

- #define **__glibcxx_want_syncbuf**
- #define **_GLIBCXX_SYNCSTREAM**

6.674.1 Detailed Description

This is a Standard C++ Library header.

6.675 experimental/system_error File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **_GLIBCXX_EXPERIMENTAL_SYSTEM_ERROR**

Variables

- template<typename _Tp>
  constexpr bool **std::experimental::is_error_code_enum_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_error_condition_enum_v**

6.675.1 Detailed Description

This is a TS C++ Library header.

6.676 system_error File Reference

Classes

- class std::error_category
- class std::error_code
- class std::error_condition
- struct std::hash< error_code >
- struct std::hash< error_condition >
- struct std::is_error_code_enum< _Tp >
- struct std::is_error_condition_enum< _Tp >
- class std::system_error

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_SYSTEM_ERROR**

Functions

- const error_category & std::generic_category () noexcept
- void **std::__adl_only::make_error_code** ()=delete
- void **std::__adl_only::make_error_condition** ()=delete
- const error_category & std::system_category () noexcept

Variables

- template<typename _Tp>
  constexpr bool **std::is_error_code_enum_v**
- template<typename _Tp>
  constexpr bool **std::is_error_condition_enum_v**

6.676.1 Detailed Description

This is a Standard C++ Library header.

6.677 text_encoding File Reference

6.677.1 Detailed Description

This is a Standard C++ Library header.

6.678 tgmath.h File Reference

6.678.1 Detailed Description

This is a Standard C++ Library header.

6.679 thread File Reference

Namespaces

- namespace std

Macros

- #define **__glibcxx_want_formatters**
- #define **__glibcxx_want_jthread**
- #define **_GLIBCXX_THREAD**

6.679.1 Detailed Description

This is a Standard C++ Library header.

6.680 bool_set File Reference

Classes

- class std::tr2::bool_set

Namespaces

- namespace std
- namespace std::tr2

Macros

- #define **_GLIBCXX_TR2_BOOL_SET**

Functions

- bool **std::tr2::certainly** (bool_set __b)
- bool **std::tr2::contains** (bool_set __s, bool_set __t)
- bool **std::tr2::equals** (bool_set __s, bool_set __t)
- bool **std::tr2::is_emptyset** (bool_set __b)
- bool **std::tr2::is_indeterminate** (bool_set __b)
- bool **std::tr2::is_singleton** (bool_set __b)
- bool_set **std::tr2::operator!=** (bool __s, bool_set __t)
- bool_set **std::tr2::operator!=** (bool_set __s, bool __t)
- bool_set **std::tr2::operator!=** (bool_set __s, bool_set __t)
- bool_set **std::tr2::operator&** (bool __s, bool_set __t)
- bool_set **std::tr2::operator&** (bool_set __s, bool __t)
- bool_set **std::tr2::operator==** (bool __s, bool_set __t)
- bool_set **std::tr2::operator==** (bool_set __s, bool __t)
- bool_set **std::tr2::operator^** (bool __s, bool_set __t)
- bool_set **std::tr2::operator^** (bool_set __s, bool __t)
- bool_set **std::tr2::operator|** (bool __s, bool_set __t)
- bool_set **std::tr2::operator|** (bool_set __s, bool __t)
- bool **std::tr2::possibly** (bool_set __b)
- bool_set **std::tr2::set_complement** (bool_set __b)
- bool_set **std::tr2::set_intersection** (bool __s, bool_set __t)
- bool_set **std::tr2::set_intersection** (bool_set __s, bool __t)

- bool_set **std::tr2::set_intersection** (bool_set __s, bool_set __t)
- bool_set **std::tr2::set_union** (bool __s, bool_set __t)
- bool_set **std::tr2::set_union** (bool_set __s, bool __t)
- bool_set **std::tr2::set_union** (bool_set __s, bool_set __t)

6.680.1 Detailed Description

This is a TR2 C++ Library header.

6.681 bool_set.tcc File Reference

Namespaces

- namespace std
- namespace std::tr2

Macros

- #define **_GLIBCXX_TR2_BOOL_SET_TCC**

6.681.1 Detailed Description

This is a TR2 C++ Library header.

6.682 dynamic_bitset File Reference

Classes

- struct std::tr2::__dynamic_bitset_base< _WordT, _Alloc >
- class std::tr2::dynamic_bitset< _WordT, _Alloc >
- class std::tr2::dynamic_bitset< _WordT, _Alloc >::reference

Namespaces

- namespace std
- namespace std::tr2

Macros

- #define **_GLIBCXX_TR2_DYNAMIC_BITSET**

Functions

- template<typename _CharT, typename _Traits, typename _WordT, typename _Alloc>
  std::basic_ostream< _CharT, _Traits > & std::tr2::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const dynamic_bitset< _WordT, _Alloc > &__x)

- template<typename _WordT, typename _Alloc>
  bool std::tr2::operator!= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)
- template<typename _WordT, typename _Alloc>
  bool std::tr2::operator<= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _↩ Alloc > &__rhs)

- template<typename _WordT, typename _Alloc>
  bool std::tr2::operator> (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)
- template<typename _WordT, typename _Alloc>
  bool std::tr2::operator>= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)

- template<typename _WordT, typename _Alloc>
  dynamic_bitset< _WordT, _Alloc > std::tr2::operator& (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)
- template<typename _WordT, typename _Alloc>
  dynamic_bitset< _WordT, _Alloc > std::tr2::operator- (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)
- template<typename _WordT, typename _Alloc>
  dynamic_bitset< _WordT, _Alloc > std::tr2::operator^ (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)
- template<typename _WordT, typename _Alloc>
  dynamic_bitset< _WordT, _Alloc > std::tr2::operator| (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)

6.682.1 Detailed Description

This is a TR2 C++ Library header.

6.683 dynamic_bitset.tcc File Reference

Namespaces

- namespace std
- namespace std::tr2

Macros

- #define **_GLIBCXX_TR2_DYNAMIC_BITSET_TCC**

Functions

- template<typename _CharT, typename _Traits, typename _WordT, typename _Alloc>
  std::basic_istream< _CharT, _Traits > & std::tr2::operator>> (std::basic_istream< _CharT, _Traits > &__is, dynamic_bitset< _WordT, _Alloc > &__x)

6.683.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <tr2/dynamic_bitset>.

6.684 experimental/tuple File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **__cpp_lib_experimental_tuple**
- #define **_GLIBCXX_EXPERIMENTAL_TUPLE**

Functions

- template<typename _Fn, typename _Tuple, std::size_t... _Idx>
  constexpr decltype(auto) **std::experimental::__apply_impl** (_Fn &&__f, _Tuple &&__t, std::index_sequence< _Idx... >)
- template<typename _Fn, typename _Tuple>
  constexpr decltype(auto) **std::experimental::apply** (_Fn &&__f, _Tuple &&__t)

Variables

- template<typename _Tp>
  constexpr size_t **std::experimental::tuple_size_v**

6.684.1 Detailed Description

This is a TS C++ Library header.

6.685 tuple File Reference

Classes

- class std::tuple< _Elements >
- class std::tuple< _T1, _T2 >
- struct std::tuple_element< __i, tuple< _Types... > >
- struct std::tuple_size< tuple< _Elements... > >
- struct std::uses_allocator< tuple< _Types... >, _Alloc >

Namespaces

- namespace std

Macros

- #define **__glibcxx_want_apply**
- #define **__glibcxx_want_constexpr_tuple**
- #define **__glibcxx_want_constrained_equality**
- #define **__glibcxx_want_make_from_tuple**
- #define **__glibcxx_want_ranges_zip**
- #define **__glibcxx_want_tuple_element_t**
- #define **__glibcxx_want_tuple_like**
- #define **__glibcxx_want_tuples_by_type**
- #define **_GLIBCXX_TUPLE**

Functions

- template<size_t __i, typename _Head, typename... _Tail>
  constexpr _Head & **std::__get_helper** (_Tuple_impl< __i, _Head, _Tail... > &__t) noexcept
- template<size_t __i, typename _Head, typename... _Tail>
  constexpr const _Head & **std::__get_helper** (const _Tuple_impl< __i, _Head, _Tail... > &__t) noexcept

- template<size_t __i, typename... _Types>
  __enable_if_t<(__i >=sizeof...(_Types))> **std::__get_helper** (const tuple< _Types... > &)=delete
- template<typename _Cat, typename _Tp, typename _Up, typename _IndexSeq>
  constexpr _Cat **std::__tuple_cmp** (const _Tp &__t, const _Up &__u, _IndexSeq __indices)
- template<typename... _Elements>
  constexpr tuple< _Elements &&... > std::forward_as_tuple (_Elements &&... __args) noexcept
- template<size_t __i, typename... _Elements>
  constexpr const __tuple_element_t< __i, tuple< _Elements... > > && std::get (const tuple< _Elements... > &&__t) noexcept
- template<size_t __i, typename... _Elements>
  constexpr const __tuple_element_t< __i, tuple< _Elements... > > & std::get (const tuple< _Elements... > &__t) noexcept
- template<size_t __i, typename... _Elements>
  constexpr __tuple_element_t< __i, tuple< _Elements... > > && std::get (tuple< _Elements... > &&__t) noexcept
- template<size_t __i, typename... _Elements>
  constexpr __tuple_element_t< __i, tuple< _Elements... > > & std::get (tuple< _Elements... > &__t) noexcept
- template<typename... _Elements>
  constexpr tuple< typename __decay_and_strip< _Elements >::__type... > std::make_tuple (_Elements &&... __args)
- template<typename... _Tps, typename... _Ups>
  requires (sizeof...(_Tps) == sizeof...(_Ups)) && (requires { typename __detail::__synth3way_t<_Tps, _Ups>; } && ...)
  constexpr common_comparison_category_t< __detail::__synth3way_t< _Tps, _Ups >... > **std::operator<=>** (const tuple< _Tps... > &__t, const tuple< _Ups... > &__u)
- template<typename... _Tps, typename... _Ups>
  requires (sizeof...(_Tps) == sizeof...(_Ups)) && (requires (const _Tps& __t, const _Ups& __u) { { __t == __u } -> __detail::__boolean_testable; } && ...)
  constexpr bool **std::operator==** (const tuple< _Tps... > &__t, const tuple< _Ups... > &__u)
- template<typename... _Elements>
  constexpr enable_if<!__and_< __is_swappable< _Elements >... >::value >::type std::swap (tuple< _Elements... > &, tuple< _Elements... > &)=delete
- template<typename... _Elements>
  constexpr enable_if< __and_< __is_swappable< _Elements >... >::value >::type std::swap (tuple< _Elements... > &__x, tuple< _Elements... > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename... _Elements>
  constexpr tuple< _Elements &... > std::tie (_Elements &... __args) noexcept
- template<typename... _UTypes>
  std::tuple (_UTypes...) -> tuple< _UTypes... >
- template<typename _Alloc, typename... _UTypes>
  std::tuple (allocator_arg_t, _Alloc, _UTypes...) -> tuple< _UTypes... >
- template<typename _Alloc, typename _T1, typename _T2>
  std::tuple (allocator_arg_t, _Alloc, pair< _T1, _T2 >) -> tuple< _T1, _T2 >
- template<typename _Alloc, typename... _UTypes>
  std::tuple (allocator_arg_t, _Alloc, tuple< _UTypes... >) -> tuple< _UTypes... >
- template<typename _T1, typename _T2>
  std::tuple (pair< _T1, _T2 >) -> tuple< _T1, _T2 >
- template<typename... _Tpls, typename = typename enable_if<__and_<__is_tuple_like<_Tpls>...>::value>::type>
  constexpr auto std::tuple_cat (_Tpls &&... __tpls) -> typename __tuple_cat_result< _Tpls... >::__type

Variables

- template<typename... _Types>
  constexpr size_t **std::tuple_size_v< const tuple< _Types... > >**

- template<typename... _Types>
  constexpr size_t **std::tuple_size_v**< **tuple**< **_Types...** > >

6.685.1 Detailed Description

This is a Standard C++ Library header.

6.686 experimental/type_traits File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **_GLIBCXX_EXPERIMENTAL_TYPE_TRAITS**
- #define **__cpp_lib_experimental_type_trait_variable_templates**
- template<typename _Tp>
  constexpr size_t **std::experimental::alignment_of_v**
- template<typename _Tp, unsigned _Idx = 0>
  constexpr size_t **std::experimental::extent_v**
- template<typename _Tp>
  constexpr bool **std::experimental::has_virtual_destructor_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_abstract_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_arithmetic_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_array_v**
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::is_assignable_v**
- template<typename _Base, typename _Derived>
  constexpr bool **std::experimental::is_base_of_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_class_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_compound_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_const_v**
- template<typename _Tp, typename... _Args>
  constexpr bool **std::experimental::is_constructible_v**
- template<typename _From, typename _To>
  constexpr bool **std::experimental::is_convertible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_copy_assignable_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_copy_constructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_default_constructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_destructible_v**

- template<typename _Tp>
  constexpr bool **std::experimental::is_empty_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_enum_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_final_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_floating_point_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_function_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_fundamental_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_integral_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_literal_type_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_lvalue_reference_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_member_function_pointer_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_member_object_pointer_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_member_pointer_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_move_assignable_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_move_constructible_v**
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::is_nothrow_assignable_v**
- template<typename _Tp, typename... _Args>
  constexpr bool **std::experimental::is_nothrow_constructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_nothrow_copy_assignable_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_nothrow_copy_constructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_nothrow_default_constructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_nothrow_destructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_nothrow_move_assignable_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_nothrow_move_constructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_null_pointer_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_object_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_pod_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_pointer_v**

- template<typename _Tp>
  constexpr bool **std::experimental::is_polymorphic_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_reference_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_rvalue_reference_v**
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::is_same_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_same_v**< **_Tp, _Tp** >
- template<typename _Tp>
  constexpr bool **std::experimental::is_scalar_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_signed_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_standard_layout_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_trivial_v**
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::is_trivially_assignable_v**
- template<typename _Tp, typename... _Args>
  constexpr bool **std::experimental::is_trivially_constructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_trivially_copy_assignable_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_trivially_copy_constructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_trivially_copyable_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_trivially_default_constructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_trivially_destructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_trivially_move_assignable_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_trivially_move_constructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_union_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_unsigned_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_void_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_volatile_v**
- template<typename _Tp>
  constexpr size_t **std::experimental::rank_v**

- #define __cpp_lib_experimental_detect
- template<typename _Default, template< typename... > class _Op, typename... _Args>
  using std::experimental::detected_or
- template<typename _Default, template< typename... > class _Op, typename... _Args>
  using std::experimental::detected_or_t

- template<template< typename... > class _Op, typename... _Args>
  using std::experimental::detected_t
- template<template< typename... > class _Op, typename... _Args>
  using std::experimental::is_detected
- template<typename _To, template< typename... > class _Op, typename... _Args>
  using std::experimental::is_detected_convertible
- template<typename _To, template< typename... > class _Op, typename... _Args>
  constexpr bool std::experimental::is_detected_convertible_v
- template<typename _Expected, template< typename... > class _Op, typename... _Args>
  using std::experimental::is_detected_exact
- template<typename _Expected, template< typename... > class _Op, typename... _Args>
  constexpr bool std::experimental::is_detected_exact_v
- template<template< typename... > class _Op, typename... _Args>
  constexpr bool std::experimental::is_detected_v
- template<typename...>
  using std::experimental::void_t

- #define **__cpp_lib_experimental_logical_traits**
- template<typename... _Bn>
  constexpr bool **std::experimental::conjunction_v**
- template<typename... _Bn>
  constexpr bool **std::experimental::disjunction_v**
- template<typename _Pp>
  constexpr bool **std::experimental::negation_v**

6.686.1 Detailed Description

This is a TS C++ Library header.
This header defines variable templates for the C++14 type traits.
Equivalent variable templates are defined in namespace `std` since C++17.

See also

variable_templates

Since

C++14

6.687 tr2/type_traits File Reference

Classes

- struct std::tr2::__reflection_typelist< _First, _Rest... >
- struct std::tr2::__reflection_typelist<>
- struct std::tr2::bases< _Tp >
- struct std::tr2::direct_bases< _Tp >

Namespaces

- namespace std
- namespace std::tr2

Macros

- #define **_GLIBCXX_TR2_TYPE_TRAITS**

6.687.1 Detailed Description

This is a TR2 C++ Library header.

6.688 type_traits File Reference

Classes

- struct std::add_const< _Tp >
- struct std::add_cv< _Tp >
- struct std::add_lvalue_reference< _Tp >
- struct std::add_pointer< _Tp >
- struct std::add_rvalue_reference< _Tp >
- struct std::add_volatile< _Tp >
- struct std::aligned_storage< _Len, _Align >
- struct std::aligned_union< _Len, _Types >
- struct std::alignment_of< _Tp >
- struct std::conditional< _Cond, _Iftrue, _Iffalse >
- struct std::decay< _Tp >
- struct std::enable_if< bool, _Tp >
- struct std::extent< typename, _Uint >
- struct std::has_virtual_destructor< _Tp >
- struct std::integral_constant< _Tp, __v >
- struct std::is_abstract< _Tp >
- struct std::is_arithmetic< _Tp >
- struct std::is_array< _Tp >
- struct std::is_assignable< _Tp, _Up >
- struct std::is_base_of< _Base, _Derived >
- struct std::is_class< _Tp >
- struct std::is_compound< _Tp >
- struct std::is_const< _Tp >
- struct std::is_constructible< _Tp, _Args >
- struct std::is_copy_assignable< _Tp >
- struct std::is_copy_constructible< _Tp >
- struct std::is_default_constructible< _Tp >
- struct std::is_destructible< _Tp >
- struct std::is_empty< _Tp >
- struct std::is_enum< _Tp >
- struct std::is_floating_point< _Tp >
- struct std::is_function< _Tp >
- struct std::is_fundamental< _Tp >
- struct std::is_integral< _Tp >
- struct std::is_layout_compatible< _Tp, _Up >
- struct std::is_literal_type< _Tp >
- struct std::is_lvalue_reference< typename >
- struct std::is_member_function_pointer< _Tp >
- struct std::is_member_object_pointer< _Tp >
- struct std::is_member_pointer< _Tp >
- struct std::is_move_assignable< _Tp >
- struct std::is_move_constructible< _Tp >
- struct std::is_nothrow_assignable< _Tp, _Up >
- struct std::is_nothrow_constructible< _Tp, _Args >

- struct std::is_nothrow_copy_assignable< _Tp >
- struct std::is_nothrow_copy_constructible< _Tp >
- struct std::is_nothrow_default_constructible< _Tp >
- struct std::is_nothrow_destructible< _Tp >
- struct std::is_nothrow_move_assignable< _Tp >
- struct std::is_nothrow_move_constructible< _Tp >
- struct std::is_object< _Tp >
- struct std::is_pod< _Tp >
- struct std::is_pointer< _Tp >
- struct std::is_pointer_interconvertible_base_of< _Base, _Derived >
- struct std::is_polymorphic< _Tp >
- struct std::is_reference< _Tp >
- struct std::is_rvalue_reference< typename >
- struct std::is_same< _Tp, _Up >
- struct std::is_scalar< _Tp >
- struct std::is_signed< _Tp >
- struct std::is_standard_layout< _Tp >
- struct std::is_trivial< _Tp >
- struct std::is_trivially_assignable< _Tp, _Up >
- struct std::is_trivially_constructible< _Tp, _Args >
- struct std::is_trivially_copy_assignable< _Tp >
- struct std::is_trivially_copy_constructible< _Tp >
- struct std::is_trivially_copyable< _Tp >
- struct std::is_trivially_default_constructible< _Tp >
- struct std::is_trivially_destructible< _Tp >
- struct std::is_trivially_move_assignable< _Tp >
- struct std::is_trivially_move_constructible< _Tp >
- struct std::is_union< _Tp >
- struct std::is_unsigned< _Tp >
- struct std::is_void< _Tp >
- struct std::is_volatile< _Tp >
- struct std::make_signed< _Tp >
- struct std::make_unsigned< _Tp >
- struct std::rank< _Tp >
- struct std::remove_all_extents< _Tp >
- struct std::remove_const< _Tp >
- struct std::remove_cv< _Tp >
- struct std::remove_extent< _Tp >
- struct std::remove_pointer< _Tp >
- struct std::remove_reference< _Tp >
- struct std::remove_volatile< _Tp >
- struct std::underlying_type< _Tp >

Namespaces

- namespace std

Macros

- #define **__glibcxx_want_bool_constant**
- #define **__glibcxx_want_bounded_array_traits**
- #define **__glibcxx_want_constant_wrapper**
- #define **__glibcxx_want_has_unique_object_representations**
- #define **__glibcxx_want_integral_constant_callable**
- #define **__glibcxx_want_is_aggregate**
- #define **__glibcxx_want_is_constant_evaluated**
- #define **__glibcxx_want_is_final**
- #define **__glibcxx_want_is_invocable**
- #define **__glibcxx_want_is_layout_compatible**
- #define **__glibcxx_want_is_nothrow_convertible**
- #define **__glibcxx_want_is_null_pointer**
- #define **__glibcxx_want_is_pointer_interconvertible**
- #define **__glibcxx_want_is_scoped_enum**
- #define **__glibcxx_want_is_swappable**
- #define **__glibcxx_want_is_virtual_base_of**
- #define **__glibcxx_want_logical_traits**
- #define **__glibcxx_want_reference_from_temporary**
- #define **__glibcxx_want_remove_cvref**
- #define **__glibcxx_want_result_of_sfinae**
- #define **__glibcxx_want_transformation_trait_aliases**
- #define **__glibcxx_want_type_identity**
- #define **__glibcxx_want_type_trait_variable_templates**
- #define **__glibcxx_want_unwrap_ref**
- #define **__glibcxx_want_void_t**
- #define **_GLIBCXX_TYPE_TRAITS**

Typedefs

- template<bool _Cond, typename _If, typename _Else>
  using **std::__conditional_t**
- template<bool _Cond, typename _Tp = void>
  using **std::__enable_if_t**
- template<typename _ToElementType, typename _FromElementType>
  using **std::__is_array_convertible**
- template<typename _Tp>
  using std::add_lvalue_reference_t
- template<typename _Tp>
  using std::add_pointer_t
- template<typename _Tp>
  using std::add_rvalue_reference_t
- template<size_t _Len, size_t _Align = __aligned_storage_default_alignment(_Len)>
  using std::aligned_storage_t
- template<size_t _Len, typename... _Types>
  using **std::aligned_union_t**
- template<typename... _Tp>
  using std::common_type_t
- template<bool _Cond, typename _Iftrue, typename _Iffalse>
  using std::conditional_t

- template<typename _Tp>
  using std::decay_t
- template<bool _Cond, typename _Tp = void>
  using std::enable_if_t
- using std::false_type
- template<typename _Tp>
  using std::make_signed_t
- template<typename _Tp>
  using std::make_unsigned_t
- template<typename _Tp>
  using std::remove_all_extents_t
- template<typename _Tp>
  using std::remove_extent_t
- template<typename _Tp>
  using std::remove_pointer_t
- template<typename _Tp>
  using std::remove_reference_t
- template<typename _Tp>
  using std::result_of_t
- using std::true_type
- template<typename _Tp>
  using std::underlying_type_t

Functions

- template<typename _Tp>
  auto std::declval () noexcept -> decltype(__declval< _Tp >(0))
- template<typename _Tp>
  constexpr _Require< __not_< __is_tuple_like< _Tp > >, is_move_constructible< _Tp >, is_move_assignable< _Tp > > std::swap (_Tp &, _Tp &) noexcept(__and_< is_nothrow_move_constructible< _Tp >, is_nothrow_move_assignable< _Tp > >::value)
- template<typename _Tp, size_t _Nm>
  constexpr __enable_if_t< __is_swappable< _Tp >::value > std::swap (_Tp(&__a)[_Nm], _Tp(&__b)[_Nm]) noexcept(__is_nothrow_swappable< _Tp >::value)

Variables

- template<typename _Tp, typename... _Args>
  constexpr bool **std::__is_nothrow_new_constructible**
- template<size_t _Len, typename... _Types>
  const size_t **std::aligned_union< _Len, _Types... >::alignment_value**

- template<typename... _Tp>
  using std::common_reference_t
- template<typename _S1, typename _S2, typename _M1, typename _M2>
  constexpr bool std::is_corresponding_member (_M1 _S1::*__m1, _M2 _S2::*__m2) noexcept
- template<typename _Tp, typename _Up>
  constexpr bool std::is_layout_compatible_v
- template<typename _Base, typename _Derived>
  constexpr bool std::is_pointer_interconvertible_base_of_v
- template<typename _Tp, typename _Mem>
  constexpr bool std::is_pointer_interconvertible_with_class (_Mem _Tp::*__mp) noexcept

6.688.1 Detailed Description

This is a Standard C++ Library header.

6.689 typeindex File Reference

Classes

- struct std::hash< type_index >
- struct std::type_index

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_TYPEINDEX**

6.689.1 Detailed Description

This is a Standard C++ Library header.

6.690 debug/unordered_map File Reference

Classes

- class std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >
- class std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >

Namespaces

- namespace std
- namespace std::__debug

Macros

- #define **_GLIBCXX_DEBUG_UNORDERED_MAP**

Functions

- template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>
  bool **std::__debug::operator==** (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>
  bool **std::__debug::operator==** (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>
  void **std::__debug::swap** (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>
  void **std::__debug::swap** (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))

- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  std::__debug::unordered_map (_InputIterator, _InputIterator, _Allocator) -> unordered_map< __iter_key↩ _t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator > >, equal_to< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  std::__debug::unordered_map (_InputIterator, _InputIterator, typename unordered_map< int, int >::size_↩ type, _Allocator) -> unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator > >, equal_to< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = ↩ _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_map (_InputIterator, _InputIterator, typename unordered_map< int, int >::size_type, _Hash, _Allocator) -> unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Hash, equal_to< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Hash = hash<__iter_key_t<_InputIterator>>, typename _Pred = equal_to<__iter_key_↩ t<_InputIterator>>, typename _Allocator = allocator<__iter_to_alloc_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_↩ Allocator>>
  std::__debug::unordered_map (_InputIterator, _InputIterator, typename unordered_map< int, int >::size_↩ type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_map< __iter_key_t< _Input↩ Iterator >, __iter_val_t< _InputIterator >, _Hash, _Pred, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_map (initializer_list< pair< _Key, _Tp > >, _Allocator) -> unordered_map< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_map (initializer_list< pair< _Key, _Tp > >, typename unordered_map< int, int >↩ ::size_type, _Allocator) -> unordered_map< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_map (initializer_list< pair< _Key, _Tp > >, typename unordered_map< int, int >↩ ::size_type, _Hash, _Allocator) -> unordered_map< _Key, _Tp, _Hash, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_map (initializer_list< pair< _Key, _Tp > >, typename unordered_map< int, int >↩ ::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_map< _Key, _Tp, _Hash, _Pred, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  std::__debug::unordered_multimap (_InputIterator, _InputIterator, _Allocator) -> unordered_multimap< ↩ __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator > >, equal_to< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  std::__debug::unordered_multimap (_InputIterator, _InputIterator, unordered_multimap< int, int >::size_type, _Allocator) -> unordered_multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator > >, equal_to< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = ↩ _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_multimap (_InputIterator, _InputIterator, unordered_multimap< int, int >::size_type, _Hash, _Allocator) -> unordered_multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, ↩ _Hash, equal_to< __iter_key_t< _InputIterator > >, _Allocator >

- template<typename _InputIterator, typename _Hash = hash<__iter_key_t<_InputIterator>>, typename _Pred = equal_to<__iter_key_t<_InputIterator>>, typename _Allocator = allocator<__iter_to_alloc_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_multimap (_InputIterator, _InputIterator, unordered_multimap< int, int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Hash, _Pred, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_multimap (initializer_list< pair< _Key, _Tp > >, _Allocator) -> unordered_multimap< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_multimap (initializer_list< pair< _Key, _Tp > >, unordered_multimap< int, int >::size_type, _Allocator) -> unordered_multimap< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_multimap (initializer_list< pair< _Key, _Tp > >, unordered_multimap< int, int >::size_type, _Hash, _Allocator) -> unordered_multimap< _Key, _Tp, _Hash, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_multimap (initializer_list< pair< _Key, _Tp > >, unordered_multimap< int, int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_multimap< _Key, _Tp, _Hash, _Pred, _Allocator >

6.690.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

6.691 experimental/unordered_map File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **_GLIBCXX_EXPERIMENTAL_UNORDERED_MAP**

Typedefs

- template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>>
  using **std::experimental::fundamentals_v2::pmr::unordered_map**
- template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>>
  using **std::experimental::fundamentals_v2::pmr::unordered_multimap**

Functions

- template<typename _Key, typename _Tp, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>
  void **std::experimental::erase_if** (unordered_map< _Key, _Tp, _Hash, _CPred, _Alloc > &__cont, _Predicate __pred)
- template<typename _Key, typename _Tp, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>
  void **std::experimental::erase_if** (unordered_multimap< _Key, _Tp, _Hash, _CPred, _Alloc > &__cont, _Predicate __pred)

6.691.1 Detailed Description

This is a TS C++ Library header.

6.692 unordered_map File Reference

Namespaces

- namespace std

Macros

- #define **__glibcxx_want_allocator_traits_is_always_equal**
- #define **__glibcxx_want_containers_ranges**
- #define **__glibcxx_want_erase_if**
- #define **__glibcxx_want_generic_unordered_lookup**
- #define **__glibcxx_want_node_extract**
- #define **__glibcxx_want_nonmember_container_access**
- #define **__glibcxx_want_tuple_like**
- #define **__glibcxx_want_unordered_map_try_emplace**
- #define **_GLIBCXX_UNORDERED_MAP**

Typedefs

- template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>, typename _Pred = std::equal_to<_Key>>
  using **std::pmr::unordered_map**
- template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>, typename _Pred = std::equal_to<_Key>>
  using **std::pmr::unordered_multimap**

Functions

- template<typename _Key, typename _Tp, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>
  unordered_map< _Key, _Tp, _Hash, _CPred, _Alloc >::size_type **std::erase_if** (unordered_map< _Key, _Tp, _Hash, _CPred, _Alloc > &__cont, _Predicate __pred)
- template<typename _Key, typename _Tp, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>
  unordered_multimap< _Key, _Tp, _Hash, _CPred, _Alloc >::size_type **std::erase_if** (unordered_multimap< ↩
  _Key, _Tp, _Hash, _CPred, _Alloc > &__cont, _Predicate __pred)

6.692.1 Detailed Description

This is a Standard C++ Library header.

6.693 debug/unordered_set File Reference

Classes

- class std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >
- class std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc >

Namespaces

- namespace std
- namespace std::__debug

Macros

- #define **_GLIBCXX_DEBUG_UNORDERED_SET**

Functions

- template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>
  bool **std::__debug::operator==** (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)
- template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>
  bool **std::__debug::operator==** (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)
- template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>
  void **std::__debug::swap** (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>
  void **std::__debug::swap** (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, unordered_set< _Value, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_multiset (_InputIterator, _InputIterator, _Allocator) -> unordered_multiset< typename iterator_traits< _InputIterator >::value_type, hash< typename iterator_traits< _InputIterator >::value_type >, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_multiset (_InputIterator, _InputIterator, unordered_multiset< int >::size_type, _Allocator) -> unordered_multiset< typename iterator_traits< _InputIterator >::value_type, hash< typename iterator_traits< _InputIterator >::value_type >, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_multiset (_InputIterator, _InputIterator, unordered_multiset< int >::size_type, _Hash, _Allocator) -> unordered_multiset< typename iterator_traits< _InputIterator >::value_type, _Hash, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Hash = hash<typename iterator_traits<_InputIterator>::value_type>, typename _Pred = equal_to<typename iterator_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_multiset (_InputIterator, _InputIterator, unordered_multiset< int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_multiset< typename iterator_traits< _InputIterator >::value_type, _Hash, _Pred, _Allocator >
- template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_multiset (initializer_list< _Tp >, _Allocator) -> unordered_multiset< _Tp, hash< _Tp >, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_multiset (initializer_list< _Tp >, unordered_multiset< int >::size_type, _Allocator) -> unordered_multiset< _Tp, hash< _Tp >, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_multiset (initializer_list< _Tp >, unordered_multiset< int >::size_type, _Hash, _Allocator) -> unordered_multiset< _Tp, _Hash, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Hash = hash<_Tp>, typename _Pred = equal_to<_Tp>, typename _Allocator = allocator<_Tp>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>

std::__debug::unordered_multiset (initializer_list< _Tp >, unordered_multiset< int >::size_type={}, _Hash=↩ _Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_multiset< _Tp, _Hash, _Pred, _Allocator >

- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  std::__debug::unordered_set (_InputIterator, _InputIterator, _Allocator) -> unordered_set< typename iterator_traits< _InputIterator >::value_type, hash< typename iterator_traits< _InputIterator >::value_type >, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  std::__debug::unordered_set (_InputIterator, _InputIterator, unordered_set< int >::size_type, _Allocator) -> unordered_set< typename iterator_traits< _InputIterator >::value_type, hash< typename iterator_traits< _↩ InputIterator >::value_type >, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = ↩ _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_set (_InputIterator, _InputIterator, unordered_set< int >::size_type, _Hash, _↩ Allocator) -> unordered_set< typename iterator_traits< _InputIterator >::value_type, _Hash, equal_to< type-name iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Hash = hash<typename iterator_traits<_InputIterator>::value_type>, typename _Pred = equal_to<typename iterator_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator_traits<_Input↩ Iterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_set (_InputIterator, _InputIterator, unordered_set< int >::size_type={}, _Hash=_↩ Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_set< typename iterator_traits< _InputIterator >::value_type, _Hash, _Pred, _Allocator >
- template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_set (initializer_list< _Tp >, _Allocator) -> unordered_set< _Tp, hash< _Tp >, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_set (initializer_list< _Tp >, unordered_set< int >::size_type, _Allocator) -> unordered_set< _Tp, hash< _Tp >, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _↩ RequireAllocator<_Allocator>>
  std::__debug::unordered_set (initializer_list< _Tp >, unordered_set< int >::size_type, _Hash, _Allocator) -> unordered_set< _Tp, _Hash, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Hash = hash<_Tp>, typename _Pred = equal_to<_Tp>, typename _Allocator = allocator<_Tp>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_↩ Allocator>>
  std::__debug::unordered_set (initializer_list< _Tp >, unordered_set< int >::size_type={}, _Hash=_Hash(), ↩ _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_set< _Tp, _Hash, _Pred, _Allocator >

6.693.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

6.694 experimental/unordered_set File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **_GLIBCXX_EXPERIMENTAL_UNORDERED_SET**

Typedefs

- template<typename _Key, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>>
  using **std::experimental::fundamentals_v2::pmr::unordered_multiset**
- template<typename _Key, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>>
  using **std::experimental::fundamentals_v2::pmr::unordered_set**

Functions

- template<typename _Key, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>
  void **std::experimental::erase_if** (unordered_multiset< _Key, _Hash, _CPred, _Alloc > &__cont, _Predicate __pred)
- template<typename _Key, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>
  void **std::experimental::erase_if** (unordered_set< _Key, _Hash, _CPred, _Alloc > &__cont, _Predicate __pred)

6.694.1 Detailed Description

This is a TS C++ Library header.

6.695 unordered_set File Reference

Namespaces

- namespace std

Macros

- #define **__glibcxx_want_allocator_traits_is_always_equal**
- #define **__glibcxx_want_containers_ranges**
- #define **__glibcxx_want_erase_if**
- #define **__glibcxx_want_generic_unordered_lookup**
- #define **__glibcxx_want_node_extract**
- #define **__glibcxx_want_nonmember_container_access**
- #define **_GLIBCXX_UNORDERED_SET**

Typedefs

- template<typename _Key, typename _Hash = std::hash<_Key>, typename _Pred = std::equal_to<_Key>>
  using **std::pmr::unordered_multiset**
- template<typename _Key, typename _Hash = std::hash<_Key>, typename _Pred = std::equal_to<_Key>>
  using **std::pmr::unordered_set**

Functions

- template<typename _Key, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>
  unordered_multiset< _Key, _Hash, _CPred, _Alloc >::size_type **std::erase_if** (unordered_multiset< _Key, _Hash, _CPred, _Alloc > &__cont, _Predicate __pred)
- template<typename _Key, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>
  unordered_set< _Key, _Hash, _CPred, _Alloc >::size_type **std::erase_if** (unordered_set< _Key, _Hash, _CPred, _Alloc > &__cont, _Predicate __pred)

6.695.1 Detailed Description

This is a Standard C++ Library header.

6.696 experimental/utility File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **_GLIBCXX_EXPERIMENTAL_UTILITY**

Typedefs

- using **std::experimental::erased_type**

6.696.1 Detailed Description

This is a TS C++ Library header.

6.697 utility File Reference

Namespaces

- namespace std

Macros

- #define **__glibcxx_want_addressof_constexpr**
- #define **__glibcxx_want_as_const**
- #define **__glibcxx_want_constexpr_algorithms**
- #define **__glibcxx_want_constexpr_utility**
- #define **__glibcxx_want_constrained_equality**
- #define **__glibcxx_want_exchange_function**
- #define **__glibcxx_want_forward_like**
- #define **__glibcxx_want_integer_comparison_functions**
- #define **__glibcxx_want_integer_sequence**
- #define **__glibcxx_want_ranges_zip**
- #define **__glibcxx_want_to_underlying**
- #define **__glibcxx_want_tuple_element_t**
- #define **__glibcxx_want_tuple_like**
- #define **__glibcxx_want_tuples_by_type**
- #define **__glibcxx_want_unreachable**
- #define **_GLIBCXX_UTILITY**

6.697.1 Detailed Description

This is a Standard C++ Library header.

6.698 valarray File Reference

Classes

- class std::valarray< _Tp >

Namespaces

- namespace std
- namespace std::__detail

Macros

- #define **_GLIBCXX_VALARRAY**

Functions

- template<class _Tp>
  const _Tp * std::begin (const valarray< _Tp > &__va) noexcept
- template<class _Tp>
  _Tp * std::begin (valarray< _Tp > &__va) noexcept
- template<class _Tp>
  const _Tp * std::end (const valarray< _Tp > &__va) noexcept
- template<class _Tp>
  _Tp * std::end (valarray< _Tp > &__va) noexcept
- template<typename _Tp, size_t _Nm>
  std::valarray (const _Tp(&)[_Nm], size_t) -> valarray< _Tp >

6.698.1 Detailed Description

This is a Standard C++ Library header.

6.699 variant File Reference

Macros

- #define **__glibcxx_want_constrained_equality**
- #define **__glibcxx_want_freestanding_variant**
- #define **__glibcxx_want_variant**
- #define **_GLIBCXX_VARIANT**

6.699.1 Detailed Description

This is the <variant> C++ Library header.

6.700 debug/vector File Reference

Classes

- class __gnu_debug::_Safe_vector< _SafeSequence, _BaseSequence >
- struct std::hash< __debug::vector< bool, _Alloc > >
- class std::__debug::vector< _Tp, _Allocator >

Namespaces

- namespace __gnu_debug
- namespace std
- namespace std::__debug
- namespace std::__detail

Macros

- #define **_GLIBCXX_DEBUG_VECTOR**

Functions

- template<typename _Tp, typename _Alloc>
  constexpr __detail::__synth3way_t< _Tp > **std::__debug::operator<=>** (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)
- template<typename _Tp, typename _Alloc>
  constexpr bool **std::__debug::operator==** (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)
- template<typename _Tp, typename _Alloc>
  constexpr void **std::__debug::swap** (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &__rhs) noexcept(/*conditional */)
- template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  std::__debug::vector (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> vector< _ValT, _Allocator >
- template<typename _Tp, typename _Allocator = allocator<_Tp>, typename = _RequireAllocator<_Allocator>>
  std::__debug::vector (size_t, _Tp, _Allocator=_Allocator()) -> vector< _Tp, _Allocator >

6.700.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

6.701 experimental/vector File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **__cpp_lib_experimental_erase_if**
- #define **_GLIBCXX_EXPERIMENTAL_VECTOR**

Typedefs

- template<typename _Tp>
  using **std::experimental::fundamentals_v2::pmr::vector**

Functions

- template<typename _Tp, typename _Alloc, typename _Up>
  void **std::experimental::erase** (vector< _Tp, _Alloc > &__cont, const _Up &__value)
- template<typename _Tp, typename _Alloc, typename _Predicate>
  void **std::experimental::erase_if** (vector< _Tp, _Alloc > &__cont, _Predicate __pred)

6.701.1 Detailed Description

This is a TS C++ Library header.

6.702 vector File Reference

Namespaces

- namespace std

Macros

- #define **__glibcxx_want_algorithm_default_value_type**
- #define **__glibcxx_want_allocator_traits_is_always_equal**
- #define **__glibcxx_want_constexpr_vector**
- #define **__glibcxx_want_containers_ranges**
- #define **__glibcxx_want_erase_if**
- #define **__glibcxx_want_incomplete_container_elements**
- #define **__glibcxx_want_nonmember_container_access**
- #define **_GLIBCXX_VECTOR**

Typedefs

- template<typename _Tp>
  using **std::pmr::vector**

6.702.1 Detailed Description

This is a Standard C++ Library header.

6.703 atomic_word.h File Reference

Macros

- #define **_GLIBCXX_READ_MEM_BARRIER**
- #define **_GLIBCXX_WRITE_MEM_BARRIER**

Typedefs

- typedef int **_Atomic_word**

6.703.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

6.704 basic_file.h File Reference

Namespaces

- namespace std

6.704.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ios>.

6.705 c++allocator.h File Reference

Namespaces

- namespace std

Typedefs

- template<typename _Tp>
  using std::__allocator_base

6.705.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

6.706 c++config.h File Reference

Namespaces

- namespace __gnu_cxx
- namespace std

Macros

- #define **__GLIBCXX__**
- #define **__glibcxx_assert**(cond)
- #define **__N**(msgid)
- #define **_GLIBCXX11_DEPRECATED**
- #define **_GLIBCXX11_DEPRECATED_SUGGEST**(ALT)
- #define **_GLIBCXX11_USE_C99_MATH**
- #define **_GLIBCXX11_USE_C99_STDIO**
- #define **_GLIBCXX11_USE_C99_STDLIB**
- #define **_GLIBCXX11_USE_C99_WCHAR**
- #define **_GLIBCXX14_DEPRECATED**
- #define **_GLIBCXX14_DEPRECATED_SUGGEST**(ALT)
- #define **_GLIBCXX17_DEPRECATED**
- #define **_GLIBCXX17_DEPRECATED_SUGGEST**(ALT)
- #define **_GLIBCXX20_DEPRECATED**
- #define **_GLIBCXX20_DEPRECATED_SUGGEST**(ALT)
- #define **_GLIBCXX23_DEPRECATED**
- #define **_GLIBCXX23_DEPRECATED_SUGGEST**(ALT)
- #define **_GLIBCXX26_CONSTEXPR**
- #define **_GLIBCXX26_DEPRECATED**
- #define **_GLIBCXX26_DEPRECATED_SUGGEST**(ALT)
- #define **_GLIBCXX98_USE_C99_COMPLEX**
- #define **_GLIBCXX98_USE_C99_MATH**
- #define **_GLIBCXX98_USE_C99_STDIO**
- #define **_GLIBCXX98_USE_C99_STDLIB**
- #define **_GLIBCXX98_USE_C99_WCHAR**
- #define **_GLIBCXX_ABI_TAG_CXX11**
- #define **_GLIBCXX_ASSERT_FAIL**(_Condition)
- #define **_GLIBCXX_ASSERTIONS**
- #define **_GLIBCXX_ATOMIC_WORD_BUILTINS**
- #define **_GLIBCXX_AUTO_CAST**(X)
- #define **_GLIBCXX_BEGIN_EXTERN_C**
- #define **_GLIBCXX_BEGIN_INLINE_ABI_NAMESPACE**(X)
- #define **_GLIBCXX_BEGIN_NAMESPACE_ALGO**
- #define **_GLIBCXX_BEGIN_NAMESPACE_CONTAINER**

- #define **_GLIBCXX_BEGIN_NAMESPACE_CXX11**
- #define **_GLIBCXX_BEGIN_NAMESPACE_LDBL**
- #define **_GLIBCXX_BEGIN_NAMESPACE_LDBL_OR_CXX11**
- #define **_GLIBCXX_BEGIN_NAMESPACE_VERSION**
- #define **_GLIBCXX_CAN_ALIGNAS_DESTRUCTIVE_SIZE**
- #define **_GLIBCXX_DARWIN_USE_64_BIT_INODE**
- #define **_GLIBCXX_DEFAULT_ABI_TAG**
- #define **_GLIBCXX_DEPRECATED**
- #define **_GLIBCXX_DEPRECATED_SUGGEST**(ALT)
- #define **_GLIBCXX_DOXYGEN_ONLY**(X)
- #define **_GLIBCXX_END_EXTERN_C**
- #define **_GLIBCXX_END_INLINE_ABI_NAMESPACE**(X)
- #define **_GLIBCXX_END_NAMESPACE_ALGO**
- #define **_GLIBCXX_END_NAMESPACE_CONTAINER**
- #define **_GLIBCXX_END_NAMESPACE_CXX11**
- #define **_GLIBCXX_END_NAMESPACE_LDBL**
- #define **_GLIBCXX_END_NAMESPACE_LDBL_OR_CXX11**
- #define **_GLIBCXX_END_NAMESPACE_VERSION**
- #define **_GLIBCXX_EXTERN_TEMPLATE**
- #define **_GLIBCXX_EXTERN_TEMPLATE**
- #define **_GLIBCXX_FAST_MATH**
- #define **_GLIBCXX_FULLY_DYNAMIC_STRING**
- #define **_GLIBCXX_HAS_BUILTIN**(B)
- #define **_GLIBCXX_HAVE___CXA_THREAD_ATEXIT_IMPL**
- #define **_GLIBCXX_HAVE_ACOSF**
- #define **_GLIBCXX_HAVE_ACOSL**
- #define **_GLIBCXX_HAVE_ALIGNED_ALLOC**
- #define **_GLIBCXX_HAVE_ARC4RANDOM**
- #define **_GLIBCXX_HAVE_ARPA_INET_H**
- #define **_GLIBCXX_HAVE_AS_SYMVER_DIRECTIVE**
- #define **_GLIBCXX_HAVE_ASINF**
- #define **_GLIBCXX_HAVE_ASINL**
- #define **_GLIBCXX_HAVE_AT_QUICK_EXIT**
- #define **_GLIBCXX_HAVE_ATAN2F**
- #define **_GLIBCXX_HAVE_ATAN2L**
- #define **_GLIBCXX_HAVE_ATANF**
- #define **_GLIBCXX_HAVE_ATANL**
- #define **_GLIBCXX_HAVE_ATOMIC_LOCK_POLICY**
- #define **_GLIBCXX_HAVE_ATTRIBUTE_VISIBILITY**
- #define **_GLIBCXX_HAVE_BUILTIN_HAS_UNIQ_OBJ_REP**
- #define **_GLIBCXX_HAVE_BUILTIN_IS_AGGREGATE**
- #define **_GLIBCXX_HAVE_BUILTIN_LAUNDER**
- #define **_GLIBCXX_HAVE_C99_FLT_EVAL_TYPES**
- #define **_GLIBCXX_HAVE_CEILF**
- #define **_GLIBCXX_HAVE_CEILL**
- #define **_GLIBCXX_HAVE_COMPLEX_H**
- #define **_GLIBCXX_HAVE_COSF**
- #define **_GLIBCXX_HAVE_COSHF**
- #define **_GLIBCXX_HAVE_COSHL**
- #define **_GLIBCXX_HAVE_COSL**
- #define **_GLIBCXX_HAVE_DECL_STRNLEN**

- #define **_GLIBCXX_HAVE_DIRENT_H**
- #define **_GLIBCXX_HAVE_DIRFD**
- #define **_GLIBCXX_HAVE_DLFCN_H**
- #define **_GLIBCXX_HAVE_ENDIAN_H**
- #define **_GLIBCXX_HAVE_EXCEPTION_PTR_SINCE_GCC46**
- #define **_GLIBCXX_HAVE_EXECINFO_H**
- #define **_GLIBCXX_HAVE_EXPF**
- #define **_GLIBCXX_HAVE_EXPL**
- #define **_GLIBCXX_HAVE_FABSF**
- #define **_GLIBCXX_HAVE_FABSL**
- #define **_GLIBCXX_HAVE_FCNTL_H**
- #define **_GLIBCXX_HAVE_FDOPENDIR**
- #define **_GLIBCXX_HAVE_FENV_H**
- #define **_GLIBCXX_HAVE_FINITE**
- #define **_GLIBCXX_HAVE_FINITEF**
- #define **_GLIBCXX_HAVE_FINITEL**
- #define **_GLIBCXX_HAVE_FLOAT_H**
- #define **_GLIBCXX_HAVE_FLOORF**
- #define **_GLIBCXX_HAVE_FLOORL**
- #define **_GLIBCXX_HAVE_FMODF**
- #define **_GLIBCXX_HAVE_FMODL**
- #define **_GLIBCXX_HAVE_FREXPF**
- #define **_GLIBCXX_HAVE_FREXPL**
- #define **_GLIBCXX_HAVE_GETENTROPY**
- #define **_GLIBCXX_HAVE_GETIPINFO**
- #define **_GLIBCXX_HAVE_GETS**
- #define **_GLIBCXX_HAVE_HYPOT**
- #define **_GLIBCXX_HAVE_HYPOTF**
- #define **_GLIBCXX_HAVE_HYPOTL**
- #define **_GLIBCXX_HAVE_ICONV**
- #define **_GLIBCXX_HAVE_INTTYPES_H**
- #define **_GLIBCXX_HAVE_IS_CONSTANT_EVALUATED**
- #define **_GLIBCXX_HAVE_ISINF**
- #define **_GLIBCXX_HAVE_ISINFF**
- #define **_GLIBCXX_HAVE_ISINFL**
- #define **_GLIBCXX_HAVE_ISNAN**
- #define **_GLIBCXX_HAVE_ISNANF**
- #define **_GLIBCXX_HAVE_ISNANL**
- #define **_GLIBCXX_HAVE_ISWBLANK**
- #define **_GLIBCXX_HAVE_LC_MESSAGES**
- #define **_GLIBCXX_HAVE_LDEXPF**
- #define **_GLIBCXX_HAVE_LDEXPL**
- #define **_GLIBCXX_HAVE_LIBINTL_H**
- #define **_GLIBCXX_HAVE_LIMIT_AS**
- #define **_GLIBCXX_HAVE_LIMIT_DATA**
- #define **_GLIBCXX_HAVE_LIMIT_FSIZE**
- #define **_GLIBCXX_HAVE_LIMIT_RSS**
- #define **_GLIBCXX_HAVE_LIMIT_VMEM**
- #define **_GLIBCXX_HAVE_LINK**
- #define **_GLIBCXX_HAVE_LINK_H**
- #define **_GLIBCXX_HAVE_LINUX_FUTEX**

- #define **_GLIBCXX_HAVE_LINUX_RANDOM_H**
- #define **_GLIBCXX_HAVE_LINUX_TYPES_H**
- #define **_GLIBCXX_HAVE_LOCALE_H**
- #define **_GLIBCXX_HAVE_LOG10F**
- #define **_GLIBCXX_HAVE_LOG10L**
- #define **_GLIBCXX_HAVE_LOGF**
- #define **_GLIBCXX_HAVE_LOGL**
- #define **_GLIBCXX_HAVE_LSEEK**
- #define **_GLIBCXX_HAVE_MBSTATE_T**
- #define **_GLIBCXX_HAVE_MEMALIGN**
- #define **_GLIBCXX_HAVE_MEMORY_H**
- #define **_GLIBCXX_HAVE_MODFF**
- #define **_GLIBCXX_HAVE_MODFL**
- #define **_GLIBCXX_HAVE_NETDB_H**
- #define **_GLIBCXX_HAVE_NETINET_IN_H**
- #define **_GLIBCXX_HAVE_NETINET_TCP_H**
- #define **_GLIBCXX_HAVE_O_NONBLOCK**
- #define **_GLIBCXX_HAVE_OPENAT**
- #define **_GLIBCXX_HAVE_POLL**
- #define **_GLIBCXX_HAVE_POLL_H**
- #define **_GLIBCXX_HAVE_POSIX_MEMALIGN**
- #define **_GLIBCXX_HAVE_POWF**
- #define **_GLIBCXX_HAVE_POWL**
- #define **_GLIBCXX_HAVE_QUICK_EXIT**
- #define **_GLIBCXX_HAVE_READLINK**
- #define **_GLIBCXX_HAVE_S_ISREG**
- #define **_GLIBCXX_HAVE_SECURE_GETENV**
- #define **_GLIBCXX_HAVE_SETENV**
- #define **_GLIBCXX_HAVE_SINCOS**
- #define **_GLIBCXX_HAVE_SINCOSF**
- #define **_GLIBCXX_HAVE_SINCOSL**
- #define **_GLIBCXX_HAVE_SINF**
- #define **_GLIBCXX_HAVE_SINHF**
- #define **_GLIBCXX_HAVE_SINHL**
- #define **_GLIBCXX_HAVE_SINL**
- #define **_GLIBCXX_HAVE_SOCKATMARK**
- #define **_GLIBCXX_HAVE_SQRTF**
- #define **_GLIBCXX_HAVE_SQRTL**
- #define **_GLIBCXX_HAVE_STACKTRACE**
- #define **_GLIBCXX_HAVE_STDALIGN_H**
- #define **_GLIBCXX_HAVE_STDBOOL_H**
- #define **_GLIBCXX_HAVE_STDINT_H**
- #define **_GLIBCXX_HAVE_STDLIB_H**
- #define **_GLIBCXX_HAVE_STRERROR_L**
- #define **_GLIBCXX_HAVE_STRERROR_R**
- #define **_GLIBCXX_HAVE_STRING_H**
- #define **_GLIBCXX_HAVE_STRINGS_H**
- #define **_GLIBCXX_HAVE_STRTOF**
- #define **_GLIBCXX_HAVE_STRTOLD**
- #define **_GLIBCXX_HAVE_STRUCT_DIRENT_D_TYPE**
- #define **_GLIBCXX_HAVE_STRXFRM_L**

- #define **_GLIBCXX_HAVE_SYMLINK**
- #define **_GLIBCXX_HAVE_SYMVER_SYMBOL_RENAMING_RUNTIME_SUPPORT**
- #define **_GLIBCXX_HAVE_SYS_IOCTL_H**
- #define **_GLIBCXX_HAVE_SYS_IPC_H**
- #define **_GLIBCXX_HAVE_SYS_MMAN_H**
- #define **_GLIBCXX_HAVE_SYS_PARAM_H**
- #define **_GLIBCXX_HAVE_SYS_PTRACE_H**
- #define **_GLIBCXX_HAVE_SYS_RESOURCE_H**
- #define **_GLIBCXX_HAVE_SYS_SEM_H**
- #define **_GLIBCXX_HAVE_SYS_SOCKET_H**
- #define **_GLIBCXX_HAVE_SYS_STAT_H**
- #define **_GLIBCXX_HAVE_SYS_STATVFS_H**
- #define **_GLIBCXX_HAVE_SYS_SYSINFO_H**
- #define **_GLIBCXX_HAVE_SYS_TIME_H**
- #define **_GLIBCXX_HAVE_SYS_TYPES_H**
- #define **_GLIBCXX_HAVE_SYS_UIO_H**
- #define **_GLIBCXX_HAVE_TANF**
- #define **_GLIBCXX_HAVE_TANHF**
- #define **_GLIBCXX_HAVE_TANHL**
- #define **_GLIBCXX_HAVE_TANL**
- #define **_GLIBCXX_HAVE_TGMATH_H**
- #define **_GLIBCXX_HAVE_TIMESPEC_GET**
- #define **_GLIBCXX_HAVE_TLS**
- #define **_GLIBCXX_HAVE_TRUNCATE**
- #define **_GLIBCXX_HAVE_UCHAR_H**
- #define **_GLIBCXX_HAVE_UNISTD_H**
- #define **_GLIBCXX_HAVE_UNLINKAT**
- #define **_GLIBCXX_HAVE_USELOCALE**
- #define **_GLIBCXX_HAVE_UTIME_H**
- #define **_GLIBCXX_HAVE_VFWSCANF**
- #define **_GLIBCXX_HAVE_VSWSCANF**
- #define **_GLIBCXX_HAVE_VWSCANF**
- #define **_GLIBCXX_HAVE_WCHAR_H**
- #define **_GLIBCXX_HAVE_WCSTOF**
- #define **_GLIBCXX_HAVE_WCTYPE_H**
- #define **_GLIBCXX_HAVE_WRITEV**
- #define **_GLIBCXX_HOSTED**
- #define **_GLIBCXX_ICONV_CONST**
- #define **_GLIBCXX_INLINE_VERSION**
- #define **_GLIBCXX_LT_OBJDIR**
- #define **_GLIBCXX_MANGLE_SIZE_T**
- #define **_GLIBCXX_NAMESPACE_CXX11**
- #define **_GLIBCXX_NAMESPACE_LDBL**
- #define **_GLIBCXX_NAMESPACE_LDBL_OR_CXX11**
- #define **_GLIBCXX_NODISCARD**
- #define **_GLIBCXX_NOEXCEPT_PARM**
- #define **_GLIBCXX_NOEXCEPT_QUAL**
- #define **_GLIBCXX_PSEUDO_VISIBILITY**(V)
- #define **_GLIBCXX_RELEASE**
- #define **_GLIBCXX_RES_LIMITS**
- #define **_GLIBCXX_STATIC_TZDATA**

- #define **_GLIBCXX_STD_A**
- #define **_GLIBCXX_STD_C**
- #define **_GLIBCXX_STDIO_EOF**
- #define **_GLIBCXX_STDIO_SEEK_CUR**
- #define **_GLIBCXX_STDIO_SEEK_END**
- #define **_GLIBCXX_SYMVER**
- #define **_GLIBCXX_SYMVER_GNU**
- #define **_GLIBCXX_SYNCHRONIZATION_HAPPENS_AFTER**(A)
- #define **_GLIBCXX_SYNCHRONIZATION_HAPPENS_BEFORE**(A)
- #define **_GLIBCXX_THROW_OR_ABORT**(_EXC)
- #define **_GLIBCXX_TXN_SAFE**
- #define **_GLIBCXX_TXN_SAFE_DYN**
- #define **_GLIBCXX_USE_ALLOCATOR_NEW**
- #define **_GLIBCXX_USE_BUILTIN_TRAIT**(BT)
- #define **_GLIBCXX_USE_C11_UCHAR_CXX11**
- #define **_GLIBCXX_USE_C99**
- #define **_GLIBCXX_USE_C99_COMPLEX**
- #define **_GLIBCXX_USE_C99_COMPLEX_TR1**
- #define **_GLIBCXX_USE_C99_CTYPE**
- #define **_GLIBCXX_USE_C99_CTYPE_TR1**
- #define **_GLIBCXX_USE_C99_FENV**
- #define **_GLIBCXX_USE_C99_FENV_TR1**
- #define **_GLIBCXX_USE_C99_INTTYPES**
- #define **_GLIBCXX_USE_C99_INTTYPES_TR1**
- #define **_GLIBCXX_USE_C99_INTTYPES_WCHAR_T**
- #define **_GLIBCXX_USE_C99_INTTYPES_WCHAR_T_TR1**
- #define **_GLIBCXX_USE_C99_MATH**
- #define **_GLIBCXX_USE_C99_MATH_FUNCS**
- #define **_GLIBCXX_USE_C99_MATH_TR1**
- #define **_GLIBCXX_USE_C99_STDINT**
- #define **_GLIBCXX_USE_C99_STDINT_TR1**
- #define **_GLIBCXX_USE_C99_STDIO**
- #define **_GLIBCXX_USE_C99_STDLIB**
- #define **_GLIBCXX_USE_C99_WCHAR**
- #define **_GLIBCXX_USE_CHDIR**
- #define **_GLIBCXX_USE_CHMOD**
- #define **_GLIBCXX_USE_CLOCK_MONOTONIC**
- #define **_GLIBCXX_USE_CLOCK_REALTIME**
- #define **_GLIBCXX_USE_CXX11_ABI**
- #define **_GLIBCXX_USE_DECIMAL_FLOAT**
- #define **_GLIBCXX_USE_DEV_RANDOM**
- #define **_GLIBCXX_USE_DUAL_ABI**
- #define **_GLIBCXX_USE_FCHMOD**
- #define **_GLIBCXX_USE_FCHMODAT**
- #define **_GLIBCXX_USE_FSEEKO_FTELLO**
- #define **_GLIBCXX_USE_GET_NPROCS**
- #define **_GLIBCXX_USE_GETCWD**
- #define **_GLIBCXX_USE_GETTIMEOFDAY**
- #define **_GLIBCXX_USE_INIT_PRIORITY_ATTRIBUTE**
- #define **_GLIBCXX_USE_LFS**
- #define **_GLIBCXX_USE_LONG_LONG**

- #define **_GLIBCXX_USE_LSTAT**
- #define **_GLIBCXX_USE_MKDIR**
- #define **_GLIBCXX_USE_NANOSLEEP**
- #define **_GLIBCXX_USE_NL_LANGINFO_L**
- #define **_GLIBCXX_USE_NLS**
- #define **_GLIBCXX_USE_PROC_SELF_STATUS**
- #define **_GLIBCXX_USE_PTHREAD_COND_CLOCKWAIT**
- #define **_GLIBCXX_USE_PTHREAD_MUTEX_CLOCKLOCK**
- #define **_GLIBCXX_USE_PTHREAD_RWLOCK_CLOCKLOCK**
- #define **_GLIBCXX_USE_RANDOM_TR1**
- #define **_GLIBCXX_USE_REALPATH**
- #define **_GLIBCXX_USE_SC_NPROCESSORS_ONLN**
- #define **_GLIBCXX_USE_SCHED_YIELD**
- #define **_GLIBCXX_USE_SENDFILE**
- #define **_GLIBCXX_USE_ST_MTIM**
- #define **_GLIBCXX_USE_STD_SPEC_FUNCS**
- #define **_GLIBCXX_USE_STRUCT_TM_TM_ZONE**
- #define **_GLIBCXX_USE_TMPNAM**
- #define **_GLIBCXX_USE_UCHAR_C8RTOMB_MBRTOC8_CXX20**
- #define **_GLIBCXX_USE_UCHAR_C8RTOMB_MBRTOC8_FCHAR8_T**
- #define **_GLIBCXX_USE_UTIME**
- #define **_GLIBCXX_USE_UTIMENSAT**
- #define **_GLIBCXX_USE_WCHAR_T**
- #define **_GLIBCXX_USE_WEAK_REF**
- #define **_GLIBCXX_VERBOSE**
- #define **_GLIBCXX_VERBOSE_ASSERT**
- #define **_GLIBCXX_VISIBILITY**(V)
- #define **_GLIBCXX_WEAK_DEFINITION**
- #define **_GLIBCXX_X86_RDRAND**
- #define **_GLIBCXX_X86_RDSEED**
- #define **_GLIBCXX_ZONEINFO_DIR**
- #define **_GTHREAD_USE_MUTEX_TIMEDLOCK**

Typedefs

- typedef decltype(nullptr) **std::nullptr_t**
- typedef __PTRDIFF_TYPE__ **std::ptrdiff_t**
- typedef __SIZE_TYPE__ **std::size_t**

Functions

- void **std::__glibcxx_assert_fail** (const char *__file, int __line, const char *__function, const char *__condition) noexcept
- constexpr bool **std::__is_constant_evaluated** () noexcept
- void **std::__terminate** () noexcept

6.706.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <version>.

6.707 c++io.h File Reference

Namespaces

- namespace std

Typedefs

- typedef FILE **std::__c_file**
- typedef __gthread_mutex_t **std::__c_lock**

6.707.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ios>.

6.708 c++locale.h File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_C_LOCALE_GNU**
- #define **_GLIBCXX_NUM_CATEGORIES**

Typedefs

- typedef __locale_t **std::__c_locale**

Functions

- int **std::__convert_from_v** (const __c_locale &__cloc, char *__out, const int __size, const char *__fmt,...)

6.708.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

6.709 c++locale_internal.h File Reference

Namespaces

- namespace std

Functions

- Catalogs & **std::get_catalogs** ()

6.709.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

6.710 parallel/compatibility.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _Tp>
  _Tp **__gnu_parallel::__add_omp** (volatile _Tp *__ptr, _Tp __addend)
- template<typename _Tp>
  bool **__gnu_parallel::__cas_omp** (volatile _Tp *__ptr, _Tp __comparand, _Tp __replacement)
- template<typename _Tp>
  bool __gnu_parallel::__compare_and_swap (volatile _Tp *__ptr, _Tp __comparand, _Tp __replacement)
- template<typename _Tp>
  _Tp __gnu_parallel::__fetch_and_add (volatile _Tp *__ptr, _Tp __addend)
- void __gnu_parallel::__yield ()

6.710.1 Detailed Description

Compatibility layer, mostly concerned with atomic operations.
This file is a GNU parallel extension to the Standard C++ Library and contains implementation details for the library's internal use.

6.711 x86_64-pc-linux-gnu/bits/compatibility.h File Reference

6.711.1 Detailed Description

This is an internal header file, included by other library sources. You should not attempt to use it directly.

6.712 cpu_defines.h File Reference

6.712.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iosfwd>.

6.713 ctype_base.h File Reference

Classes

- struct std::ctype_base

Namespaces

- namespace std

6.713.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

6.714 ctype_inline.h File Reference

Namespaces

- namespace std

6.714.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

6.715 cxxabi_tweaks.h File Reference

Macros

- #define **_GLIBCXX_CXA_VEC_CTOR_RETURN**(x)
- #define **_GLIBCXX_GUARD_BIT**
- #define **_GLIBCXX_GUARD_PENDING_BIT**
- #define **_GLIBCXX_GUARD_SET**(x)
- #define **_GLIBCXX_GUARD_TEST**(x)
- #define **_GLIBCXX_GUARD_WAITING_BIT**

Typedefs

- typedef void **__cxxabiv1::__cxa_cdtor_return_type**
- typedef void **__cxxabiv1::__cxa_vec_ctor_return_type**

Variables

- __extension__ typedef int **__cxxabiv1::__guard**

6.715.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <cxxabi.h>.

6.716 error_constants.h File Reference

Namespaces

- namespace std

Enumerations

- enum class **errc** {
  address_family_not_supported , **address_in_use** , **address_not_available** , **already_connected** ,
  argument_list_too_long , **argument_out_of_domain** , **bad_address** , **bad_file_descriptor** ,
  broken_pipe , **connection_aborted** , **connection_already_in_progress** , **connection_refused** ,
  connection_reset , **cross_device_link** , **destination_address_required** , **device_or_resource_busy** ,
  directory_not_empty , **executable_format_error** , **file_exists** , **file_too_large** ,
  filename_too_long , **function_not_supported** , **host_unreachable** , **illegal_byte_sequence** ,
  inappropriate_io_control_operation , **interrupted** , **invalid_argument** , **invalid_seek** ,
  io_error , **is_a_directory** , **message_size** , **network_down** ,
  network_reset , **network_unreachable** , **no_buffer_space** , **no_child_process** ,
  no_lock_available , **no_message** , **no_protocol_option** , **no_space_on_device** ,
  no_such_device_or_address , **no_such_device** , **no_such_file_or_directory** , **no_such_process** ,
  not_a_directory , **not_a_socket** , **not_connected** , **not_enough_memory** ,
  operation_in_progress , **operation_not_permitted** , **operation_not_supported** , **operation_would_block** ,
  permission_denied , **protocol_not_supported** , **read_only_file_system** , **resource_deadlock_would_occur**
  ,
  resource_unavailable_try_again , **result_out_of_range** , **timed_out** , **too_many_files_open_in_system** ,
  too_many_files_open , **too_many_links** , **too_many_symbolic_link_levels** , **wrong_protocol_type** }

6.716.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <system_error>.

6.717 extc++.h File Reference

6.717.1 Detailed Description

This is an implementation file for a precompiled header.

6.718 messages_members.h File Reference

Namespaces

- namespace std

6.718.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

6.719 opt_random.h File Reference

Namespaces

- namespace std

6.719.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <random>.

6.720 os_defines.h File Reference

Macros

- #define **__NO_CTYPE**
- #define **_GLIBCXX_MAY_HAVE___CXA_THREAD_ATEXIT_IMPL**
- #define **_GLIBCXX_NO_OBSOLETE_ISINF_ISNAN_DYNAMIC**

6.720.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iosfwd>.

6.721 stdc++.h File Reference

6.721.1 Detailed Description

This is an implementation file for a precompiled header.

6.722 stdtr1c++.h File Reference

6.722.1 Detailed Description

This is an implementation file for a precompiled header.

6.723 time_members.h File Reference

Namespaces

- namespace std

6.723.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

6.724 compare File Reference

Classes

- struct std::compare_three_way_result< _Tp, _Up >

Namespaces

- namespace std
- namespace std::__detail

Typedefs

- template<typename _Tp, typename _Up>
  using **std::__detail::__cmp3way_res_t**
- template<typename... _Ts>
  using **std::common_comparison_category_t**
- template<typename _Tp, typename _Up = _Tp>
  using std::compare_three_way_result_t
- using **std::__cmp_cat::type**

Enumerations

- enum class **_Ord** : type { **equivalent** , **less** , **greater** , **unordered** }

Functions

- template<typename... _Ts>
  constexpr auto **std::__detail::__common_cmp_cat** ()
- template<typename _Ordering>
  constexpr _Ordering **std::__cmp_cat::__make** (_Ord __o) noexcept
- template<typename _Ordering>
  constexpr _Ord **std::__cmp_cat::__ord** (_Ordering __o) noexcept
- constexpr bool **std::is_eq** (partial_ordering __cmp) noexcept
- constexpr bool **std::is_gt** (partial_ordering __cmp) noexcept
- constexpr bool **std::is_gteq** (partial_ordering __cmp) noexcept
- constexpr bool **std::is_lt** (partial_ordering __cmp) noexcept
- constexpr bool **std::is_lteq** (partial_ordering __cmp) noexcept
- constexpr bool **std::is_neq** (partial_ordering __cmp) noexcept

Variables

- template<typename _Tp>
  constexpr unsigned **std::__detail::__cmp_cat_id**
- template<> constexpr unsigned **std::__detail::__cmp_cat_id< partial_ordering >**
- template<> constexpr unsigned **std::__detail::__cmp_cat_id< strong_ordering >**
- template<> constexpr unsigned **std::__detail::__cmp_cat_id< weak_ordering >**
- constexpr __compare::_Partial_fallback **std::compare_partial_order_fallback**
- constexpr __compare::_Strong_fallback **std::compare_strong_order_fallback**
- constexpr __compare::_Weak_fallback **std::compare_weak_order_fallback**
- constexpr __compare::_Partial_order **std::partial_order**
- constexpr __compare::_Strong_order **std::strong_order**
- constexpr __compare::_Weak_order **std::weak_order**

6.724.1 Detailed Description

This is a Standard C++ Library header.

6.725 cxxabi.h File Reference

Classes

- class __gnu_cxx::recursive_init_error

Namespaces

- namespace __gnu_cxx
- namespace abi

Typedefs

- typedef __cxa_cdtor_return_type(* **__cxxabiv1::__cxa_cdtor_type**) (void *)

Functions

- __cxa_dependent_exception * **__cxxabiv1::__cxa_allocate_dependent_exception** () noexcept
- int **__cxxabiv1::__cxa_atexit** (void(*)(void *), void *, void *) noexcept
- void **__cxxabiv1::__cxa_bad_cast** ()
- void **__cxxabiv1::__cxa_bad_typeid** ()
- void * **__cxxabiv1::__cxa_begin_catch** (void *) noexcept
- std::type_info * **__cxxabiv1::__cxa_current_exception_type** () noexcept
- void **__cxxabiv1::__cxa_deleted_virtual** (void)
- char * __cxxabiv1::__cxa_demangle (const char *__mangled_name, char *__output_buffer, size_t *__length, int *__status)
- void **__cxxabiv1::__cxa_end_catch** ()
- void **__cxxabiv1::__cxa_finalize** (void *)
- void **__cxxabiv1::__cxa_free_dependent_exception** (__cxa_dependent_exception *) noexcept
- void **__cxxabiv1::__cxa_free_exception** (void *) noexcept
- void * **__cxxabiv1::__cxa_get_exception_ptr** (void *) noexcept
- __cxa_eh_globals * **__cxxabiv1::__cxa_get_globals** () noexcept
- __cxa_eh_globals * **__cxxabiv1::__cxa_get_globals_fast** () noexcept
- void **__cxxabiv1::__cxa_guard_abort** (__guard *) noexcept
- int **__cxxabiv1::__cxa_guard_acquire** (__guard *)
- void **__cxxabiv1::__cxa_guard_release** (__guard *) noexcept

- void **__cxxabiv1::__cxa_pure_virtual** (void)
- void **__cxxabiv1::__cxa_rethrow** ()
- int **__cxxabiv1::__cxa_thread_atexit** (void(∗)(void ∗), void ∗, void ∗) noexcept
- void **__cxxabiv1::__cxa_throw** (void ∗, std::type_info ∗, void(∗)(void ∗))
- void **__cxxabiv1::__cxa_throw_bad_array_new_length** ()
- __cxa_vec_ctor_return_type **__cxxabiv1::__cxa_vec_cctor** (void ∗__dest_array, void ∗__src_array, size_t __element_count, size_t __element_size, __cxa_cdtor_return_type(∗__constructor)(void ∗, void ∗), __cxa_cdtor_type __destructor)
- void **__cxxabiv1::__cxa_vec_cleanup** (void ∗__array_address, size_t __element_count, size_t __s, __cxa_cdtor_type __destructor) noexcept
- __cxa_vec_ctor_return_type **__cxxabiv1::__cxa_vec_ctor** (void ∗__array_address, size_t __element_count, size_t __element_size, __cxa_cdtor_type __constructor, __cxa_cdtor_type __destructor)
- void **__cxxabiv1::__cxa_vec_delete** (void ∗__array_address, size_t __element_size, size_t __padding_size, __cxa_cdtor_type __destructor)
- void **__cxxabiv1::__cxa_vec_delete2** (void ∗__array_address, size_t __element_size, size_t __padding_size, __cxa_cdtor_type __destructor, void(∗__dealloc)(void ∗))
- void **__cxxabiv1::__cxa_vec_delete3** (void ∗__array_address, size_t __element_size, size_t __padding_size, __cxa_cdtor_type __destructor, void(∗__dealloc)(void ∗, size_t))
- void **__cxxabiv1::__cxa_vec_dtor** (void ∗__array_address, size_t __element_count, size_t __element_size, __cxa_cdtor_type __destructor)
- void ∗ **__cxxabiv1::__cxa_vec_new** (size_t __element_count, size_t __element_size, size_t __padding_size, __cxa_cdtor_type __constructor, __cxa_cdtor_type __destructor)
- void ∗ **__cxxabiv1::__cxa_vec_new2** (size_t __element_count, size_t __element_size, size_t __padding_size, __cxa_cdtor_type __constructor, __cxa_cdtor_type __destructor, void ∗(∗__alloc)(size_t), void(∗__dealloc)(void ∗))
- void ∗ **__cxxabiv1::__cxa_vec_new3** (size_t __element_count, size_t __element_size, size_t __padding_size, __cxa_cdtor_type __constructor, __cxa_cdtor_type __destructor, void ∗(∗__alloc)(size_t), void(∗__dealloc)(void ∗, size_t))
- void ∗ **__cxxabiv1::__dynamic_cast** (const void ∗__src_ptr, const __class_type_info ∗__src_type, const __class_type_info ∗__dst_type, ptrdiff_t __src2dst)

6.725.1 Detailed Description

The header provides an interface to the C++ ABI.

6.725.2 Function Documentation

__cxa_demangle()

```
char * __cxxabiv1::__cxa_demangle (
            const char * __mangled_name,
            char * __output_buffer,
            size_t * __length,
            int * __status)
```

Demangling routine. ABI-mandated entry point in the C++ runtime library for demangling.

Parameters

__mangled_name	A NUL-terminated character string containing the name to be demangled.
__output_buffer	A region of memory, allocated with malloc, of ∗*__length* bytes, into which the demangled name is stored. If *__output_buffer* is not long enough, it is expanded using realloc. *__output_buffer* may instead be null; in that case, the demangled name is placed in a region of memory allocated with malloc.

Parameters

__length	If *__length* is non-null, the length of the buffer containing the demangled name is placed in *__length*.
__status	If *__status* is non-null, *__status* is set to one of the following values: 0: The demangling operation succeeded. -1: A memory allocation failure occurred. -2: *mangled_name* is not a valid name under the C++ ABI mangling rules. -3: One of the arguments is invalid.

Returns

A pointer to the start of the NUL-terminated demangled name, or a null pointer if the demangling fails. The caller is responsible for deallocating this memory using `free`.

The demangling is performed using the C++ ABI mangling rules, with GNU extensions. For example, this function is used in __gnu_cxx::__verbose_terminate_handler.
See https://gcc.gnu.org/onlinedocs/libstdc++/manual/ext_demangling.html for other examples of use.

Note

The same demangling functionality is available via libiberty (<libiberty/demangle.h> and libiberty.a) in GCC 3.1 and later, but that requires explicit installation (-enable-install-libiberty) and uses a different API, although the ABI is unchanged.

6.726 exception File Reference

Classes

- class std::bad_exception

Namespaces

- namespace __gnu_cxx
- namespace std

Typedefs

- typedef void(* std::terminate_handler) ()
- typedef void(* std::unexpected_handler) ()

Functions

- void __gnu_cxx::__verbose_terminate_handler ()
- terminate_handler std::get_terminate () noexcept
- unexpected_handler std::get_unexpected () noexcept
- terminate_handler std::set_terminate (terminate_handler) noexcept
- unexpected_handler std::set_unexpected (unexpected_handler) noexcept
- void std::terminate () noexcept
- bool std::uncaught_exception () noexcept
- void std::unexpected ()

6.726.1 Detailed Description

This is a Standard C++ Library header.

6.727 initializer_list File Reference

Classes

- class std::initializer_list< _E >

Namespaces

- namespace std

6.727.1 Detailed Description

This is a Standard C++ Library header.

6.728 new File Reference

Classes

- class std::bad_alloc
- struct std::destroying_delete_t

Namespaces

- namespace std

Macros

- #define **__glibcxx_want_constexpr_new**
- #define **__glibcxx_want_destroying_delete**
- #define **__glibcxx_want_hardware_interference_size**
- #define **__glibcxx_want_launder**
- #define **_GLIBCXX_PLACEMENT_CONSTEXPR**
- #define **_NEW**

Typedefs

- typedef void(* std::new_handler) ()

Enumerations

- enum class **align_val_t** : size_t

Functions

- new_handler std::get_new_handler () noexcept
- void **operator delete** (void *) noexcept
- void **operator delete** (void *, const std::nothrow_t &) noexcept
- void **operator delete** (void *, std::align_val_t) noexcept
- void **operator delete** (void *, std::align_val_t, const std::nothrow_t &) noexcept
- void **operator delete** (void *, std::size_t) noexcept
- void **operator delete** (void *, std::size_t, std::align_val_t) noexcept
- void **operator delete** (void *, void *) noexcept
- void **operator delete[ ]** (void *) noexcept
- void **operator delete[ ]** (void *, const std::nothrow_t &) noexcept
- void **operator delete[ ]** (void *, std::align_val_t) noexcept

- void **operator delete[ ]** (void *, std::align_val_t, const std::nothrow_t &) noexcept
- void **operator delete[ ]** (void *, std::size_t) noexcept
- void **operator delete[ ]** (void *, std::size_t, std::align_val_t) noexcept
- void **operator delete[ ]** (void *, void *) noexcept
- void * operator new (std::size_t)
- void * **operator new** (std::size_t, const std::nothrow_t &) noexcept
- void * **operator new** (std::size_t, std::align_val_t)
- void * **operator new** (std::size_t, std::align_val_t, const std::nothrow_t &) noexcept
- void * **operator new** (std::size_t, void *__p) noexcept
- void * **operator new[ ]** (std::size_t)
- void * **operator new[ ]** (std::size_t, const std::nothrow_t &) noexcept
- void * **operator new[ ]** (std::size_t, std::align_val_t)
- void * **operator new[ ]** (std::size_t, std::align_val_t, const std::nothrow_t &) noexcept
- void * **operator new[ ]** (std::size_t, void *__p) noexcept
- new_handler std::set_new_handler (new_handler) throw ()

Variables

- constexpr destroying_delete_t std::destroying_delete
- const nothrow_t **std::nothrow**

6.728.1 Detailed Description

This is a Standard C++ Library header.
The header `new` defines several functions to manage dynamic memory and handling memory allocation errors; see https://gcc.gnu.org/onlinedocs/libstdc++/manual/dynamic_memory.html for more.

6.728.2 Function Documentation

operator new()

```
void * operator new (
            std::size_t )  [nodiscard]
```

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

6.729 typeinfo File Reference

Classes

- class std::bad_cast
- class std::bad_typeid
- class std::type_info

Namespaces

- namespace std

Macros

- #define **__glibcxx_want_constexpr_typeinfo**
- #define **__GXX_MERGED_TYPEINFO_NAMES**
- #define **__GXX_TYPEINFO_EQUALITY_INLINE**
- #define **_TYPEINFO**

6.729.1 Detailed Description

This is a Standard C++ Library header.

Index

_AlgorithmStrategy
__gnu_parallel, 423
_BALLOC_ALIGN_BYTES
bitmap_allocator.h, 3607
_BinIndex
__gnu_parallel, 422
_Bit_scan_forward
__gnu_cxx, 394
_CASable
__gnu_parallel, 422
_CASable_bits
__gnu_parallel, 462
_CASable_mask
__gnu_parallel, 462
_Construct
std, 587
_DRandomShufflingGlobalData
__gnu_parallel::_DRandomShufflingGlobalData< _RAIter >, 872
_Destroy
std, 587
_Destroy_n
std, 587
_Distance_precision
__gnu_debug, 412
_FindAlgorithm
__gnu_parallel, 423
_Find_first
SGI, 152
_Find_next
SGI, 152
_GLIBCXX_BAL_QUICKSORT
features.h, 3729
_GLIBCXX_CALL
compiletime_settings.h, 3727
_GLIBCXX_DEBUG_VERIFY_AT_F
macros.h, 3584
_GLIBCXX_DEQUE_BUF_SIZE
stl_deque.h, 3484
_GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS
features.h, 3729
_GLIBCXX_FIND_EQUAL_SPLIT
features.h, 3729
_GLIBCXX_FIND_GROWING_BLOCKS
features.h, 3729
_GLIBCXX_MERGESORT
features.h, 3730
_GLIBCXX_PARALLEL_ASSERTIONS
compiletime_settings.h, 3728
_GLIBCXX_PARALLEL_CONDITION
settings.h, 3745
_GLIBCXX_PARALLEL_LENGTH
multiway_merge.h, 3737
_GLIBCXX_QUICKSORT
features.h, 3730
_GLIBCXX_RANDOM_SHUFFLE_CONSIDER_L1
compiletime_settings.h, 3728
_GLIBCXX_RANDOM_SHUFFLE_CONSIDER_TLB
compiletime_settings.h, 3728
_GLIBCXX_SCALE_DOWN_FPU
compiletime_settings.h, 3728
_GLIBCXX_TREE_DYNAMIC_BALANCING
features.h, 3730
_GLIBCXX_TREE_FULL_COPY
features.h, 3730
_GLIBCXX_TREE_INITIAL_SPLITTING
features.h, 3730
_GLIBCXX_VERBOSE_LEVEL
compiletime_settings.h, 3728
_GLIBCXX_VOLATILE
partition.h, 3741
queue.h, 3742
_GuardedIterator
__gnu_parallel::_GuardedIterator< _RAIter, _Compare >, 886
_LoserTreeBase
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >, 918
_M_allocate_and_copy
std::vector< _Tp, _Alloc >, 3295
std::vector< bool, _Alloc >, 3316
_M_allocate_single_object
__gnu_cxx::bitmap_allocator< _Tp >, 1758
_M_attach
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 963
__gnu_debug::_Safe_iterator_base, 972
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 978
__gnu_debug::_Safe_local_iterator_base, 986
_M_attach_single
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 964
__gnu_debug::_Safe_iterator_base, 972
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 978, 979
__gnu_debug::_Safe_local_iterator_base, 986
_M_attached_to
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 964
__gnu_debug::_Safe_iterator_base, 972

__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 979
__gnu_debug::_Safe_local_iterator_base, 986
_M_before_dereferenceable
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 964
_M_begin
__gnu_parallel::_Piece< _DifferenceTp >, 942
_M_bin_proc
__gnu_parallel::_DRandomShufflingGlobalData< _RAIter >, 872
_M_bins_begin
__gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >, 874
_M_buf
__gnu_cxx::enc_filebuf< _CharT >, 2099
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 3007
std::basic_filebuf< _CharT, _Traits >, 1108
_M_buf_locale
__gnu_cxx::enc_filebuf< _CharT >, 2099
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 3007
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3025
std::basic_filebuf< _CharT, _Traits >, 1108
std::basic_streambuf< _CharT, _Traits >, 1472
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1667
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3344
_M_buf_size
__gnu_cxx::enc_filebuf< _CharT >, 2099
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 3007
std::basic_filebuf< _CharT, _Traits >, 1108
_M_can_compare
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 964
__gnu_debug::_Safe_iterator_base, 972
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 979
__gnu_debug::_Safe_local_iterator_base, 986
_M_clear
__gnu_cxx::free_list, 2160
_M_comp
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >, 919
_M_const_iterators
__gnu_debug::_Safe_forward_list< _SafeSequence >, 959
__gnu_debug::_Safe_node_sequence< _Sequence >, 991
__gnu_debug::_Safe_sequence< _Sequence >, 994
__gnu_debug::_Safe_sequence_base, 997
__gnu_debug::_Safe_unordered_container< _Container >, 1000
__gnu_debug::_Safe_unordered_container_base, 1004
_M_const_local_iterators
__gnu_debug::_Safe_unordered_container< _Container >, 1000
__gnu_debug::_Safe_unordered_container_base, 1004
_M_create_node
std::list< _Tp, _Alloc >, 2332
_M_create_pback
__gnu_cxx::enc_filebuf< _CharT >, 2086
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2992
std::basic_filebuf< _CharT, _Traits >, 1093
_M_deallocate_single_object
__gnu_cxx::bitmap_allocator< _Tp >, 1758
_M_dereferenceable
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 964
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 979
_M_destroy_pback
__gnu_cxx::enc_filebuf< _CharT >, 2086
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2992
std::basic_filebuf< _CharT, _Traits >, 1093
_M_detach
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 965
__gnu_debug::_Safe_iterator_base, 972
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 979
__gnu_debug::_Safe_local_iterator_base, 987
_M_detach_all
__gnu_debug::_Safe_forward_list< _SafeSequence >, 959
__gnu_debug::_Safe_node_sequence< _Sequence >, 990
__gnu_debug::_Safe_sequence< _Sequence >, 993
__gnu_debug::_Safe_sequence_base, 996
__gnu_debug::_Safe_unordered_container< _Container >, 999
__gnu_debug::_Safe_unordered_container_base, 1003
_M_detach_single
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 965
__gnu_debug::_Safe_iterator_base, 972
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 979
__gnu_debug::_Safe_local_iterator_base, 987
_M_detach_singular
__gnu_debug::_Safe_forward_list< _SafeSequence >, 959
__gnu_debug::_Safe_node_sequence< _Sequence >, 990

__gnu_debug::_Safe_sequence< _Sequence >, 993
__gnu_debug::_Safe_sequence_base, 996
__gnu_debug::_Safe_unordered_container< _Container >, 999
__gnu_debug::_Safe_unordered_container_base, 1003
_M_dist
__gnu_parallel::_DRandomShufflingGlobalData< _RAIter >, 872
_M_elements_leftover
__gnu_parallel::_QSBThreadLocal< _RAIter >, 950
_M_end
__gnu_parallel::_Piece< _DifferenceTp >, 942
_M_ext_buf
__gnu_cxx::enc_filebuf< _CharT >, 2100
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 3007
std::basic_filebuf< _CharT, _Traits >, 1108
_M_ext_buf_size
__gnu_cxx::enc_filebuf< _CharT >, 2100
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 3008
std::basic_filebuf< _CharT, _Traits >, 1108
_M_ext_next
__gnu_cxx::enc_filebuf< _CharT >, 2100
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 3008
std::basic_filebuf< _CharT, _Traits >, 1108
_M_fill_initialize
std::deque< _Tp, _Alloc >, 2036
_M_finish_iterator
__gnu_parallel::__accumulate_selector< _It >, 730
__gnu_parallel::__adjacent_difference_selector< _It >, 731
__gnu_parallel::__count_if_selector< _It, _Diff >, 747
__gnu_parallel::__count_selector< _It, _Diff >, 749
__gnu_parallel::__fill_selector< _It >, 765
__gnu_parallel::__for_each_selector< _It >, 769
__gnu_parallel::__generate_selector< _It >, 771
__gnu_parallel::__generic_for_each_selector< _It >, 773
__gnu_parallel::__identity_selector< _It >, 774
__gnu_parallel::__inner_product_selector< _It, _It2, _Tp >, 776
__gnu_parallel::__replace_if_selector< _It, _Op, _Tp >, 801
__gnu_parallel::__replace_selector< _It, _Tp >, 803
__gnu_parallel::__transform1_selector< _It >, 805
__gnu_parallel::__transform2_selector< _It >, 806
_M_first
__gnu_parallel::_Job< _DifferenceTp >, 901
_M_first_insert
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >, 919
_M_gcount
std::basic_fstream< _CharT, _Traits >, 1155
std::basic_ifstream< _CharT, _Traits >, 1197
std::basic_iostream< _CharT, _Traits >, 1267
std::basic_istream< _CharT, _Traits >, 1304
std::basic_istream< _CharT, _Traits >::sentry, 2894
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1343
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1712
_M_get
__gnu_cxx::free_list, 2160
_M_get_mutex
__gnu_debug::_Safe_forward_list< _SafeSequence >, 959
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 965
__gnu_debug::_Safe_iterator_base, 973
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 979
__gnu_debug::_Safe_local_iterator_base, 987
__gnu_debug::_Safe_node_sequence< _Sequence >, 990
__gnu_debug::_Safe_sequence< _Sequence >, 993
__gnu_debug::_Safe_sequence_base, 996
__gnu_debug::_Safe_unordered_container< _Container >, 999
__gnu_debug::_Safe_unordered_container_base, 1003
_M_get_result
std::__basic_future< _Res >, 740
std::future< _Res >, 2173
std::future< _Res & >, 2175
std::future< void >, 2177
std::shared_future< _Res >, 2953
std::shared_future< _Res & >, 2955
std::shared_future< void >, 2958
_M_getloc
std::basic_fstream< _CharT, _Traits >, 1120
std::basic_ifstream< _CharT, _Traits >, 1171
std::basic_ios< _CharT, _Traits >, 1209
std::basic_iostream< _CharT, _Traits >, 1232
std::basic_istream< _CharT, _Traits >, 1278
std::basic_istream< _CharT, _Traits >::sentry, 2871
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1318
std::basic_ofstream< _CharT, _Traits >, 1356
std::basic_ostream< _CharT, _Traits >, 1389
std::basic_ostream< _CharT, _Traits >::sentry, 2904
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1421
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1678
std::ios_base, 2261

_M_global
__gnu_parallel::_QSBThreadLocal< _RAIter >, 950
_M_in_beg
__gnu_cxx::enc_filebuf< _CharT >, 2100
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 3008
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3025
std::basic_filebuf< _CharT, _Traits >, 1108
std::basic_streambuf< _CharT, _Traits >, 1472
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1667
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3344
_M_in_cur
__gnu_cxx::enc_filebuf< _CharT >, 2100
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 3008
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3026
std::basic_filebuf< _CharT, _Traits >, 1108
std::basic_streambuf< _CharT, _Traits >, 1473
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1667
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3344
_M_in_end
__gnu_cxx::enc_filebuf< _CharT >, 2100
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 3008
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3026
std::basic_filebuf< _CharT, _Traits >, 1109
std::basic_streambuf< _CharT, _Traits >, 1473
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1668
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3344
_M_in_same_bucket
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 979
_M_incrementable
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 965
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 980
_M_initial
__gnu_parallel::_QSBThreadLocal< _RAIter >, 950
_M_initialize_map
std::_Deque_base< _Tp, _Alloc >, 869
_M_insert
__gnu_cxx::free_list, 2160
_M_invalidate
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 965
__gnu_debug::_Safe_iterator_base, 973
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 980
__gnu_debug::_Safe_local_iterator_base, 987
_M_invalidate_all
__gnu_debug::_Safe_sequence< _Sequence >, 993
__gnu_debug::_Safe_sequence_base, 996
__gnu_debug::_Safe_unordered_container< _Container >, 999
__gnu_debug::_Safe_unordered_container_base, 1004
_M_invalidate_if
__gnu_debug::_Safe_forward_list< _SafeSequence >, 959
__gnu_debug::_Safe_node_sequence< _Sequence >, 990
__gnu_debug::_Safe_sequence< _Sequence >, 993
__gnu_debug::_Safe_unordered_container< _Container >, 999
_M_invalidate_local_if
__gnu_debug::_Safe_unordered_container< _Container >, 1000
_M_is_before_begin
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 965
_M_is_begin
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 965
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 980
_M_is_beginnest
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 966
_M_is_end
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 966
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 980
_M_iterators
__gnu_debug::_Safe_forward_list< _SafeSequence >, 959
__gnu_debug::_Safe_node_sequence< _Sequence >, 991
__gnu_debug::_Safe_sequence< _Sequence >, 994
__gnu_debug::_Safe_sequence_base, 997
__gnu_debug::_Safe_unordered_container< _Container >, 1000
__gnu_debug::_Safe_unordered_container_base, 1004
_M_key
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser, 912
_M_last
__gnu_parallel::_Job< _DifferenceTp >, 901
_M_leftover_parts
__gnu_parallel::_QSBThreadLocal< _RAIter >, 950
_M_load
__gnu_parallel::_Job< _DifferenceTp >, 901
_M_local_iterators

__gnu_debug::_Safe_unordered_container< _Container >, 1001
__gnu_debug::_Safe_unordered_container_base, 1004
_M_log_k
__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >, 915
__gnu_parallel::_LoserTree< false, _Tp, _Compare >, 917
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >, 919
_M_losers
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >, 919
_M_mode
__gnu_cxx::enc_filebuf< _CharT >, 2100
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 3008
std::basic_filebuf< _CharT, _Traits >, 1109
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1668
_M_new_elements_at_back
std::deque< _Tp, _Alloc >, 2036
_M_new_elements_at_front
std::deque< _Tp, _Alloc >, 2036
_M_next
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 968
__gnu_debug::_Safe_iterator_base, 973
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 983
__gnu_debug::_Safe_local_iterator_base, 988
_M_num_bins
__gnu_parallel::_DRandomShufflingGlobalData< _RAIter >, 872
_M_num_bits
__gnu_parallel::_DRandomShufflingGlobalData< _RAIter >, 872
_M_num_threads
__gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >, 874
__gnu_parallel::_PMWMSSortingData< _RAIter >, 944
__gnu_parallel::_QSBThreadLocal< _RAIter >, 950
_M_offsets
__gnu_parallel::_PMWMSSortingData< _RAIter >, 944
_M_out_beg
__gnu_cxx::enc_filebuf< _CharT >, 2100
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 3008
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3026
std::basic_filebuf< _CharT, _Traits >, 1109
std::basic_streambuf< _CharT, _Traits >, 1473
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1668
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3344
_M_out_cur
__gnu_cxx::enc_filebuf< _CharT >, 2100
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 3008
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3026
std::basic_filebuf< _CharT, _Traits >, 1109
std::basic_streambuf< _CharT, _Traits >, 1473
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1668
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3344
_M_out_end
__gnu_cxx::enc_filebuf< _CharT >, 2101
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 3008
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3026
std::basic_filebuf< _CharT, _Traits >, 1109
std::basic_streambuf< _CharT, _Traits >, 1473
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1668
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3344
_M_pback
__gnu_cxx::enc_filebuf< _CharT >, 2101
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 3009
std::basic_filebuf< _CharT, _Traits >, 1109
_M_pback_cur_save
__gnu_cxx::enc_filebuf< _CharT >, 2101
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 3009
std::basic_filebuf< _CharT, _Traits >, 1109
_M_pback_end_save
__gnu_cxx::enc_filebuf< _CharT >, 2101
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 3009
std::basic_filebuf< _CharT, _Traits >, 1110
_M_pback_init
__gnu_cxx::enc_filebuf< _CharT >, 2101
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 3009
std::basic_filebuf< _CharT, _Traits >, 1110
_M_pieces
__gnu_parallel::_PMWMSSortingData< _RAIter >, 944
_M_pop_back_aux
std::deque< _Tp, _Alloc >, 2037
_M_pop_front_aux
std::deque< _Tp, _Alloc >, 2037
_M_prior
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 968
__gnu_debug::_Safe_iterator_base, 973
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 983
__gnu_debug::_Safe_local_iterator_base, 988
_M_push_back_aux
std::deque< _Tp, _Alloc >, 2037
_M_push_front_aux
std::deque< _Tp, _Alloc >, 2037
_M_range_check
std::deque< _Tp, _Alloc >, 2037

std::vector< _Tp, _Alloc >, 3295
std::vector< bool, _Alloc >, 3316
_M_range_initialize
std::deque< _Tp, _Alloc >, 2037, 2038
_M_reading
__gnu_cxx::enc_filebuf< _CharT >, 2101
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 3009
std::basic_filebuf< _CharT, _Traits >, 1110
_M_reallocate_map
std::deque< _Tp, _Alloc >, 2038
_M_requested_size
__gnu_cxx::_Temporary_buffer< _ForwardIterator, _Tp >, 1019
__gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >, 3049
std::_Temporary_buffer< _ForwardIterator, _Tp >, 1021
_M_reserve_elements_at_back
std::deque< _Tp, _Alloc >, 2038
_M_reserve_elements_at_front
std::deque< _Tp, _Alloc >, 2039
_M_reserve_map_at_back
std::deque< _Tp, _Alloc >, 2039
_M_reserve_map_at_front
std::deque< _Tp, _Alloc >, 2039
_M_reset
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 966
__gnu_debug::_Safe_iterator_base, 973
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 980
__gnu_debug::_Safe_local_iterator_base, 987
_M_revalidate_singular
__gnu_debug::_Safe_forward_list< _SafeSequence >, 959
__gnu_debug::_Safe_node_sequence< _Sequence >, 990
__gnu_debug::_Safe_sequence< _Sequence >, 993
__gnu_debug::_Safe_sequence_base, 997
__gnu_debug::_Safe_unordered_container< _Container >, 1000
__gnu_debug::_Safe_unordered_container_base, 1004
_M_samples
__gnu_parallel::_PMWMSSortingData< _RAIter >, 944
_M_sd
__gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >, 874
_M_seed
__gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >, 874
_M_sequence
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 969
__gnu_debug::_Safe_iterator_base, 974
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 983
__gnu_debug::_Safe_local_iterator_base, 988
_M_sequential_algorithm
__gnu_parallel::__adjacent_find_selector, 732
__gnu_parallel::__find_first_of_selector< _FIterator >, 766
__gnu_parallel::__find_if_selector, 767
__gnu_parallel::__mismatch_selector, 779
_M_set_buffer
__gnu_cxx::enc_filebuf< _CharT >, 2087
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2992
std::basic_filebuf< _CharT, _Traits >, 1093
_M_set_node
std::_Deque_iterator< _Tp, _Ref, _Ptr >, 871
_M_singular
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 966
__gnu_debug::_Safe_iterator_base, 973
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 980
__gnu_debug::_Safe_local_iterator_base, 987
_M_source
__gnu_parallel::_DRandomShufflingGlobalData< _RAIter >, 873
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser, 912
__gnu_parallel::_PMWMSSortingData< _RAIter >, 945
_M_starts
__gnu_parallel::_DRandomShufflingGlobalData< _RAIter >, 873
__gnu_parallel::_PMWMSSortingData< _RAIter >, 945
_M_sup
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser, 912
_M_swap
__gnu_debug::_Safe_forward_list< _SafeSequence >, 959
__gnu_debug::_Safe_node_sequence< _Sequence >, 990
__gnu_debug::_Safe_sequence< _Sequence >, 993
__gnu_debug::_Safe_sequence_base, 997
__gnu_debug::_Safe_unordered_container< _Container >, 1000
__gnu_debug::_Safe_unordered_container_base, 1004
_M_temporaries
__gnu_parallel::_DRandomShufflingGlobalData<

_RAIter >, 873
_M_temporary
__gnu_parallel::_PMWMSSortingData< _RAIter >, 945
_M_transfer_from_if
__gnu_debug::_Safe_forward_list< _SafeSequence >, 959
__gnu_debug::_Safe_node_sequence< _Sequence >, 990
__gnu_debug::_Safe_sequence< _Sequence >, 994
_M_unlink
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 966
__gnu_debug::_Safe_iterator_base, 973
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 980
__gnu_debug::_Safe_local_iterator_base, 987
_M_use_pointer
__gnu_parallel::_LoserTreeTraits< _Tp >, 926
_M_value_initialized
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 966
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 981
_M_version
__gnu_debug::_Safe_forward_list< _SafeSequence >, 960
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 969
__gnu_debug::_Safe_iterator_base, 974
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 983
__gnu_debug::_Safe_local_iterator_base, 988
__gnu_debug::_Safe_node_sequence< _Sequence >, 991
__gnu_debug::_Safe_sequence< _Sequence >, 994
__gnu_debug::_Safe_sequence_base, 997
__gnu_debug::_Safe_unordered_container< _Container >, 1001
__gnu_debug::_Safe_unordered_container_base, 1005
_M_w
std::_Base_bitset< 0 >, 861
std::_Base_bitset< 1 >, 863
std::_Base_bitset< _Nw >, 859
std::tr2::__dynamic_bitset_base< _WordT, _Alloc >, 764
_MultiwayMergeAlgorithm
__gnu_parallel, 423
_Opcode
Base and Implementation Classes, 272
_Parallelism
__gnu_parallel, 423
_PartialSumAlgorithm
__gnu_parallel, 423
_Piece
__gnu_parallel::_QSBThreadLocal< _RAIter >, 950
_PseudoSequence
__gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >, 948
_QSBThreadLocal
__gnu_parallel::_QSBThreadLocal< _RAIter >, 950
_RandomNumber
__gnu_parallel::_RandomNumber, 951
_RestrictedBoundedConcurrentQueue
__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >, 955
_S_constant
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 967
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 981
_Safe_iterator
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 962, 963
_Safe_iterator_base
__gnu_debug::_Safe_iterator_base, 971
_Safe_local_iterator
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 977
_Safe_local_iterator_base
__gnu_debug::_Safe_local_iterator_base, 985
_SequenceIndex
__gnu_parallel, 422
_SortAlgorithm
__gnu_parallel, 423
_SplittingAlgorithm
__gnu_parallel, 423
_Temporary_buffer
__gnu_cxx::_Temporary_buffer< _ForwardIterator, _Tp >, 1019
std::_Temporary_buffer< _ForwardIterator, _Tp >, 1021
_ThreadIndex
__gnu_parallel, 422
_TokenT
std::__detail::_Scanner< _CharT >, 1008
_Unchecked_flip
SGI, 152
_Unchecked_reset
SGI, 152
_Unchecked_set
SGI, 153
_Unchecked_test
SGI, 153
__addressof

Utilities, 298
__allocator_base
Allocators, 324
__attribute
std::map< _Key, _Tp, _Compare, _Alloc >, 2386
__base
__gnu_debug, 412
__begin1_iterator
__gnu_parallel::__inner_product_selector< _It, _It2, _Tp >, 776
__begin2_iterator
__gnu_parallel::__inner_product_selector< _It, _It2, _Tp >, 776
__bins_end
__gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >, 873
__bit_allocate
__gnu_cxx::__detail, 404
__bit_free
__gnu_cxx::__detail, 404
__calc_borders
__gnu_parallel, 423
__check_singular
__gnu_debug, 412
__check_singular_aux
__gnu_debug, 412
__check_string
__gnu_debug, 412
__compare_and_swap
__gnu_parallel, 424
__cpp_lib_experimental_detect
Detection idiom, 285
__ctype_type
std::basic_ios< _CharT, _Traits >, 1207
__cxa_demangle
cxxabi.h, 3803
__cxxabiv1::__forced_unwind, 769
__decode2
__gnu_parallel, 424
__delete_min_insert
__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >, 914
__gnu_parallel::_LoserTree< false, _Tp, _Compare >, 916
__determine_samples
__gnu_parallel, 424
__encode2
__gnu_parallel, 425
__equally_split
__gnu_parallel, 425
__equally_split_point
__gnu_parallel, 426
__fetch_and_add
__gnu_parallel, 426
__find_if_not
std, 581
__find_if_not_n
std, 581
__find_template
__gnu_parallel, 426–428
__for_each_template_random_access
__gnu_parallel, 429
__for_each_template_random_access_ed
__gnu_parallel, 430
__for_each_template_random_access_omp_loop
__gnu_parallel, 430
__for_each_template_random_access_omp_loop_static
__gnu_parallel, 431
__for_each_template_random_access_workstealing
__gnu_parallel, 431
__foreign_iterator_aux2
__gnu_debug, 413
__from_chars_alnum
std::__detail, 656
__from_chars_pow2_base
std::__detail, 656
__gcd
std, 581
__gen_two_uniform_ints
std, 581
__genrand_bits
__gnu_parallel::_RandomNumber, 952
__get_distance
__gnu_debug, 413
__get_min_source
__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >, 914
__gnu_parallel::_LoserTree< false, _Tp, _Compare >, 916
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >, 919
__get_num_threads
__gnu_parallel::balanced_quicksort_tag, 1086
__gnu_parallel::balanced_tag, 1087
__gnu_parallel::default_parallel_tag, 2023
__gnu_parallel::exact_tag, 2115
__gnu_parallel::multiway_mergesort_exact_tag, 2524
__gnu_parallel::multiway_mergesort_sampling_tag, 2525
__gnu_parallel::multiway_mergesort_tag, 2526
__gnu_parallel::omp_loop_static_tag, 2683
__gnu_parallel::omp_loop_tag, 2684
__gnu_parallel::parallel_tag, 2720
__gnu_parallel::quicksort_tag, 2779
__gnu_parallel::sampling_tag, 2858
__gnu_parallel::unbalanced_tag, 3141
__glibcxx_check_erase

macros.h, 3582
__glibcxx_check_erase_after
macros.h, 3582
__glibcxx_check_erase_range
macros.h, 3582
__glibcxx_check_erase_range_after
macros.h, 3582
__glibcxx_check_heap_pred
macros.h, 3582
__glibcxx_check_insert
macros.h, 3582
__glibcxx_check_insert_after
macros.h, 3582
__glibcxx_check_insert_range
macros.h, 3583
__glibcxx_check_insert_range_after
macros.h, 3583
__glibcxx_check_partitioned_lower
macros.h, 3583
__glibcxx_check_partitioned_lower_pred
macros.h, 3583
__glibcxx_check_partitioned_upper_pred
macros.h, 3583
__glibcxx_check_sorted_pred
macros.h, 3584
__gnu_cxx, 377
_Bit_scan_forward, 394
__int_traits, 394
__static_pointer_cast, 394
iota, 394
operator!=, 395
operator<, 398
operator<=, 399, 400
operator>, 401, 402
operator>=, 402, 403
operator+, 396, 397
operator==, 400, 401
swap, 403
__gnu_cxx::_Caster< _ToType >, 865
__gnu_cxx::_Char_types< _CharT >, 865
__gnu_cxx::_ExtPtr_allocator< _Tp >, 878
__gnu_cxx::_Invalid_type, 891
__gnu_cxx::_Pointer_adapter< _Storage_policy >, 945
__gnu_cxx::_Relative_pointer_impl< _Tp >, 952
__gnu_cxx::_Relative_pointer_impl< const _Tp >, 954
__gnu_cxx::_Std_pointer_impl< _Tp >, 1017
__gnu_cxx::_Temporary_buffer< _ForwardIterator, _Tp >, 1018
_M_requested_size, 1019
_Temporary_buffer, 1019
begin, 1019
end, 1019
size, 1019
__gnu_cxx::_Unqualified_type< _Tp >, 1021
__gnu_cxx::__alloc_traits< _Alloc, typename >, 732
allocate, 735, 736
const_void_pointer, 734
construct, 736
deallocate, 737
destroy, 737
is_always_equal, 734
max_size, 738
propagate_on_container_copy_assignment, 734
propagate_on_container_move_assignment, 734
propagate_on_container_swap, 735
select_on_container_copy_construction, 738
void_pointer, 735
__gnu_cxx::__common_pool_policy< _PoolTp, _Thread >, 746
__gnu_cxx::__detail, 404
__bit_allocate, 404
__bit_free, 404
__num_bitmaps, 404
__num_blocks, 405
__gnu_cxx::__detail::_Bitmap_counter< _Tp >, 863
__gnu_cxx::__detail::_Ffit_finder< _Tp >, 879
__gnu_cxx::__detail::__mini_vector< _Tp >, 777
__gnu_cxx::__mt_alloc< _Tp, _Poolp >, 780
__gnu_cxx::__mt_alloc_base< _Tp >, 781
__gnu_cxx::__per_type_pool_policy< _Tp, _PoolTp, _Thread >, 790
__gnu_cxx::__pool< _Thread >, 791
__gnu_cxx::__pool< false >, 791
__gnu_cxx::__pool< true >, 792
__gnu_cxx::__pool_alloc< _Tp >, 793
__gnu_cxx::__pool_alloc_base, 794
__gnu_cxx::__pool_base, 795
__gnu_cxx::__rc_string_base< _CharT, _Traits, _Alloc >, 796
__gnu_cxx::__scoped_lock, 803
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 807
__versa_string, 811–813
~__versa_string, 814
append, 814–816
assign, 816–819
at, 819, 820
back, 820
begin, 821
c_str, 821
capacity, 821
cbegin, 821
cend, 821
clear, 822
compare, 822–824
copy, 825
crbegin, 825
crend, 825

data, 826
empty, 826
end, 826
erase, 826, 827
find, 827–829
find_first_not_of, 829, 830
find_first_of, 831, 832
find_last_not_of, 832–834
find_last_of, 834, 835
front, 836
get_allocator, 836
insert, 836–840
length, 841
max_size, 841
npos, 858
operator+=, 841, 843
operator=, 844, 845
operator[], 845
pop_back, 846
push_back, 846
rbegin, 846
rend, 847
replace, 847–853
reserve, 853
resize, 854
rfind, 855, 856
shrink_to_fit, 856
size, 856
substr, 857
swap, 857
__gnu_cxx::allocator< _Tp >, 1027
__gnu_cxx::annotate_base, 1063
__gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >, 1732
argument_type, 1733
result_type, 1733
__gnu_cxx::bitmap_allocator< _Tp >, 1757
_M_allocate_single_object, 1758
_M_deallocate_single_object, 1758
__gnu_cxx::char_traits< _CharT >, 1790
__gnu_cxx::character< _Value, _Int, _St >, 1796
__gnu_cxx::condition_base, 1877
__gnu_cxx::constant_binary_fun< _Result, _Arg1, _Arg2 >, 1885
__gnu_cxx::constant_unary_fun< _Result, _Argument >, 1886
__gnu_cxx::constant_void_fun< _Result >, 1887
__gnu_cxx::debug_allocator< _Alloc >, 2013
__gnu_cxx::enc_filebuf< _CharT >, 2083
_M_buf, 2099
_M_buf_locale, 2099
_M_buf_size, 2099
_M_create_pback, 2086
_M_destroy_pback, 2086
_M_ext_buf, 2100
_M_ext_buf_size, 2100
_M_ext_next, 2100
_M_in_beg, 2100
_M_in_cur, 2100
_M_in_end, 2100
_M_mode, 2100
_M_out_beg, 2100
_M_out_cur, 2100
_M_out_end, 2101
_M_pback, 2101
_M_pback_cur_save, 2101
_M_pback_end_save, 2101
_M_pback_init, 2101
_M_reading, 2101
_M_set_buffer, 2087
close, 2087
eback, 2087
egptr, 2087
epptr, 2087
gbump, 2088
getloc, 2088
gptr, 2088
imbue, 2088
in_avail, 2089
is_open, 2089
open, 2089, 2090
overflow, 2090
pbackfail, 2091
pbase, 2091
pbump, 2092
pptr, 2092
pubimbue, 2092
pubseekoff, 2092
pubseekpos, 2093
pubsetbuf, 2093
pubsync, 2093
sbumpc, 2093
seekoff, 2093
seekpos, 2094
setbuf, 2094
setg, 2094
setp, 2095
sgetc, 2095
sgetn, 2095
showmanyc, 2096
snextc, 2096
sputbackc, 2096
sputc, 2097
sputn, 2097
sungetc, 2097
sync, 2097
uflow, 2098
underflow, 2098

xsgetn, 2098
xsputn, 2099
__gnu_cxx::encoding_char_traits< _CharT >, 2102
__gnu_cxx::encoding_state, 2103
__gnu_cxx::equal_to< _Tp >, 2108
first_argument_type, 2108
result_type, 2108
second_argument_type, 2108
__gnu_cxx::forced_error, 2134
what, 2135
__gnu_cxx::free_list, 2159
_M_clear, 2160
_M_get, 2160
_M_insert, 2160
__gnu_cxx::hash_map< _Key, _Tp, _HashFn, _EqualKey, _Alloc >, 2227
__gnu_cxx::hash_multimap< _Key, _Tp, _HashFn, _EqualKey, _Alloc >, 2229
__gnu_cxx::hash_multiset< _Value, _HashFcn, _EqualKey, _Alloc >, 2230
__gnu_cxx::hash_set< _Value, _HashFcn, _EqualKey, _Alloc >, 2232
__gnu_cxx::limit_condition, 2317
__gnu_cxx::limit_condition::always_adjustor, 1062
__gnu_cxx::limit_condition::limit_adjustor, 2317
__gnu_cxx::limit_condition::never_adjustor, 2534
__gnu_cxx::malloc_allocator< _Tp >, 2376
__gnu_cxx::new_allocator< _Tp >, 2534
__gnu_cxx::pair< _T1, _T2 >, 2708
first, 2711
first_type, 2710
pair, 2710, 2711
second, 2711
second_type, 2710
swap, 2711
__gnu_cxx::project1st< _Arg1, _Arg2 >, 2770
first_argument_type, 2770
result_type, 2770
second_argument_type, 2770
__gnu_cxx::project2nd< _Arg1, _Arg2 >, 2770
first_argument_type, 2771
result_type, 2771
second_argument_type, 2771
__gnu_cxx::random_condition, 2780
__gnu_cxx::random_condition::always_adjustor, 1063
__gnu_cxx::random_condition::group_adjustor, 2202
__gnu_cxx::random_condition::never_adjustor, 2534
__gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc >, 2797
__gnu_cxx::recursive_init_error, 2811
what, 2811
__gnu_cxx::rope< _CharT, _Alloc >, 2838
__gnu_cxx::select1st< _Pair >, 2861
argument_type, 2862
result_type, 2862
__gnu_cxx::select2nd< _Pair >, 2862
argument_type, 2862
result_type, 2862
__gnu_cxx::slist< _Tp, _Alloc >, 2977
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2988
_M_buf, 3007
_M_buf_locale, 3007
_M_buf_size, 3007
_M_create_pback, 2992
_M_destroy_pback, 2992
_M_ext_buf, 3007
_M_ext_buf_size, 3008
_M_ext_next, 3008
_M_in_beg, 3008
_M_in_cur, 3008
_M_in_end, 3008
_M_mode, 3008
_M_out_beg, 3008
_M_out_cur, 3008
_M_out_end, 3008
_M_pback, 3009
_M_pback_cur_save, 3009
_M_pback_end_save, 3009
_M_pback_init, 3009
_M_reading, 3009
_M_set_buffer, 2992
~stdio_filebuf, 2992
close, 2992
eback, 2993
egptr, 2993
epptr, 2993
fd, 2993
file, 2994
gbump, 2994
getloc, 2994
gptr, 2994
imbue, 2995
in_avail, 2995
is_open, 2995
open, 2995–2997
overflow, 2997
pbackfail, 2998
pbase, 2998
pbump, 2998
pptr, 2999
pubimbue, 2999
pubseekoff, 2999
pubseekpos, 3000
pubsetbuf, 3000
pubsync, 3000
sbumpc, 3000
seekoff, 3000
seekpos, 3001

setbuf, 3001
setg, 3001
setp, 3002
sgetc, 3002
sgetn, 3002
showmanyc, 3003
snextc, 3003
sputbackc, 3003
sputc, 3004
sputn, 3004
stdio_filebuf, 2991
sungetc, 3005
sync, 3005
uflow, 3005
underflow, 3005
xsgetn, 3006
xsputn, 3006
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3010
_M_buf_locale, 3025
_M_in_beg, 3025
_M_in_cur, 3026
_M_in_end, 3026
_M_out_beg, 3026
_M_out_cur, 3026
_M_out_end, 3026
eback, 3012
egptr, 3012
epptr, 3013
file, 3013
gbump, 3013
getloc, 3013
gptr, 3014
imbue, 3014
in_avail, 3014
overflow, 3015
pbackfail, 3015
pbase, 3016
pbump, 3016
pptr, 3016
pubimbue, 3016
pubseekoff, 3017
pubseekpos, 3017
pubsetbuf, 3017
pubsync, 3018
sbumpc, 3018
seekoff, 3018
seekpos, 3018
setbuf, 3019
setg, 3019
setp, 3019
sgetc, 3020
sgetn, 3020
showmanyc, 3020
snextc, 3021
sputbackc, 3021
sputc, 3021
sputn, 3023
sungetc, 3023
sync, 3023
uflow, 3024
underflow, 3024
xsgetn, 3024
xsputn, 3025
__gnu_cxx::subtractive_rng, 3043
argument_type, 3044
operator(), 3044
result_type, 3044
subtractive_rng, 3044
__gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >, 3047
_M_requested_size, 3049
~temporary_buffer, 3048
begin, 3049
end, 3049
size, 3049
temporary_buffer, 3048
__gnu_cxx::throw_allocator_base< _Tp, _Cond >, 3053
__gnu_cxx::throw_allocator_limit< _Tp >, 3054
__gnu_cxx::throw_allocator_random< _Tp >, 3056
__gnu_cxx::throw_value_base< _Cond >, 3058
__gnu_cxx::throw_value_limit, 3059
__gnu_cxx::throw_value_random, 3060
__gnu_cxx::typelist, 405
apply_generator, 405
__gnu_cxx::unary_compose< _Operation1, _Operation2 >, 3137
argument_type, 3138
result_type, 3138
__gnu_debug, 405
_Distance_precision, 412
__base, 412
__check_singular, 412
__check_singular_aux, 412
__check_string, 412
__foreign_iterator_aux2, 413
__get_distance, 413
__valid_range, 413
__valid_range_aux, 414
u16string, 411
u32string, 411
__gnu_debug::_After_nth_from< _Iterator >, 858
__gnu_debug::_BeforeBeginHelper< _Sequence >, 863
__gnu_debug::_Equal_to< _Type >, 874
__gnu_debug::_Not_equal_to< _Type >, 940
__gnu_debug::_Safe_container< _SafeContainer, _Alloc, _SafeBase, _IsCxx11AllocatorAware >, 956
__gnu_debug::_Safe_forward_list< _SafeSequence >, 958

_M_const_iterators, 959
_M_detach_all, 959
_M_detach_singular, 959
_M_get_mutex, 959
_M_invalidate_if, 959
_M_iterators, 959
_M_revalidate_singular, 959
_M_swap, 959
_M_transfer_from_if, 959
_M_version, 960
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 960
_M_attach, 963
_M_attach_single, 964
_M_attached_to, 964
_M_before_dereferenceable, 964
_M_can_compare, 964
_M_dereferenceable, 964
_M_detach, 965
_M_detach_single, 965
_M_get_mutex, 965
_M_incrementable, 965
_M_invalidate, 965
_M_is_before_begin, 965
_M_is_begin, 965
_M_is_beginnest, 966
_M_is_end, 966
_M_next, 968
_M_prior, 968
_M_reset, 966
_M_sequence, 969
_M_singular, 966
_M_unlink, 966
_M_value_initialized, 966
_M_version, 969
_S_constant, 967
_Safe_iterator, 962, 963
base, 967
operator _Iterator, 967
operator++, 967, 968
operator->, 968
operator=, 968
operator*, 967
__gnu_debug::_Safe_iterator_base, 969
_M_attach, 972
_M_attach_single, 972
_M_attached_to, 972
_M_can_compare, 972
_M_detach, 972
_M_detach_single, 972
_M_get_mutex, 973
_M_invalidate, 973
_M_next, 973
_M_prior, 973
_M_reset, 973
_M_sequence, 974
_M_singular, 973
_M_unlink, 973
_M_version, 974
_Safe_iterator_base, 971
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 974
_M_attach, 978
_M_attach_single, 978, 979
_M_attached_to, 979
_M_can_compare, 979
_M_dereferenceable, 979
_M_detach, 979
_M_detach_single, 979
_M_get_mutex, 979
_M_in_same_bucket, 979
_M_incrementable, 980
_M_invalidate, 980
_M_is_begin, 980
_M_is_end, 980
_M_next, 983
_M_prior, 983
_M_reset, 980
_M_sequence, 983
_M_singular, 980
_M_unlink, 980
_M_value_initialized, 981
_M_version, 983
_S_constant, 981
_Safe_local_iterator, 977
base, 981
bucket, 981
operator _Iterator, 981
operator++, 982
operator->, 982
operator=, 982
operator*, 981
__gnu_debug::_Safe_local_iterator_base, 983
_M_attach, 986
_M_attach_single, 986
_M_attached_to, 986
_M_can_compare, 986
_M_detach, 987
_M_detach_single, 987
_M_get_mutex, 987
_M_invalidate, 987
_M_next, 988
_M_prior, 988
_M_reset, 987
_M_sequence, 988
_M_singular, 987
_M_unlink, 987
_M_version, 988

_Safe_local_iterator_base, 985
__gnu_debug::_Safe_node_sequence< _Sequence >, 988
_M_const_iterators, 991
_M_detach_all, 990
_M_detach_singular, 990
_M_get_mutex, 990
_M_invalidate_if, 990
_M_iterators, 991
_M_revalidate_singular, 990
_M_swap, 990
_M_transfer_from_if, 990
_M_version, 991
__gnu_debug::_Safe_sequence< _Sequence >, 991
_M_const_iterators, 994
_M_detach_all, 993
_M_detach_singular, 993
_M_get_mutex, 993
_M_invalidate_all, 993
_M_invalidate_if, 993
_M_iterators, 994
_M_revalidate_singular, 993
_M_swap, 993
_M_transfer_from_if, 994
_M_version, 994
__gnu_debug::_Safe_sequence_base, 994
_M_const_iterators, 997
_M_detach_all, 996
_M_detach_singular, 996
_M_get_mutex, 996
_M_invalidate_all, 996
_M_iterators, 997
_M_revalidate_singular, 997
_M_swap, 997
_M_version, 997
~_Safe_sequence_base, 996
__gnu_debug::_Safe_unordered_container< _Container >, 997
_M_const_iterators, 1000
_M_const_local_iterators, 1000
_M_detach_all, 999
_M_detach_singular, 999
_M_get_mutex, 999
_M_invalidate_all, 999
_M_invalidate_if, 999
_M_invalidate_local_if, 1000
_M_iterators, 1000
_M_local_iterators, 1001
_M_revalidate_singular, 1000
_M_swap, 1000
_M_version, 1001
__gnu_debug::_Safe_unordered_container_base, 1001
_M_const_iterators, 1004
_M_const_local_iterators, 1004
_M_detach_all, 1003
_M_detach_singular, 1003
_M_get_mutex, 1003
_M_invalidate_all, 1004
_M_iterators, 1004
_M_local_iterators, 1004
_M_revalidate_singular, 1004
_M_swap, 1004
_M_version, 1005
~_Safe_unordered_container_base, 1003
__gnu_debug::_Safe_vector< _SafeSequence, _BaseSequence >, 1005
__gnu_debug::_Sequence_traits< _Sequence >, 1008
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1473
__resize_and_overwrite, 1481
append, 1481, 1482
assign, 1483, 1484
at, 1485–1487
back, 1488
capacity, 1488
compare, 1488–1494
copy, 1494
data, 1495
empty, 1495
erase, 1495, 1496
find, 1497–1501
find_first_not_of, 1501–1505
find_first_of, 1506–1510
find_last_not_of, 1510–1514
find_last_of, 1514–1518
front, 1519
get_allocator, 1519
insert, 1519–1525
length, 1526
max_size, 1526
npos, 1545
operator+=, 1526
operator[], 1526, 1527
replace, 1527–1536
reserve, 1537, 1538
resize, 1539
rfind, 1539–1543
size, 1544
substr, 1544
swap, 1544
__gnu_debug::type_info, 3132
~type_info, 3133
before, 3133
name, 3133
__gnu_internal, 414
__gnu_parallel, 414
_AlgorithmStrategy, 423
_BinIndex, 422

_CASable, 422
_CASable_bits, 462
_CASable_mask, 462
_FindAlgorithm, 423
_MultiwayMergeAlgorithm, 423
_Parallelism, 423
_PartialSumAlgorithm, 423
_SequenceIndex, 422
_SortAlgorithm, 423
_SplittingAlgorithm, 423
_ThreadIndex, 422
__calc_borders, 423
__compare_and_swap, 424
__decode2, 424
__determine_samples, 424
__encode2, 425
__equally_split, 425
__equally_split_point, 426
__fetch_and_add, 426
__find_template, 426–428
__for_each_template_random_access, 429
__for_each_template_random_access_ed, 430
__for_each_template_random_access_omp_loop, 430
__for_each_template_random_access_omp_loop_static, 431
__for_each_template_random_access_workstealing, 431
__is_sorted, 432
__median_of_three_iterators, 432
__merge_advance, 433
__merge_advance_movc, 433
__merge_advance_usual, 434
__parallel_merge_advance, 435
__parallel_nth_element, 436
__parallel_partial_sort, 436
__parallel_partial_sum, 437
__parallel_partial_sum_basecase, 437
__parallel_partial_sum_linear, 437
__parallel_partition, 438
__parallel_random_shuffle, 438
__parallel_random_shuffle_drs, 439
__parallel_random_shuffle_drs_pu, 439
__parallel_sort, 440–444
__parallel_sort_qs, 445
__parallel_sort_qs_conquer, 445
__parallel_sort_qs_divide, 445
__parallel_sort_qsb, 446
__parallel_unique_copy, 446, 447
__qsb_conquer, 447
__qsb_divide, 448
__qsb_local_sort_with_helping, 448
__random_number_pow2, 449
__rd_log2, 449
__round_up_to_pow2, 449
__search_template, 449
__sequential_multiway_merge, 450
__sequential_random_shuffle, 450
__shrink, 451
__shrink_and_double, 451
__yield, 452
list_partition, 452
max, 452
min, 452
multiseq_partition, 453
multiseq_selection, 453
multiway_merge, 454
multiway_merge_3_variant, 455
multiway_merge_4_variant, 456
multiway_merge_exact_splitting, 456
multiway_merge_loser_tree, 457
multiway_merge_loser_tree_sentinel, 457
multiway_merge_loser_tree_unguarded, 458
multiway_merge_sampling_splitting, 458
multiway_merge_sentinels, 459
parallel_balanced, 423
parallel_multiway_merge, 460
parallel_omp_loop, 423
parallel_omp_loop_static, 423
parallel_sort_mwms, 461
parallel_sort_mwms_pu, 461
parallel_taskqueue, 423
parallel_unbalanced, 423
sequential, 423
__gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >, 873
_M_bins_begin, 874
_M_num_threads, 874
_M_sd, 874
_M_seed, 874
__bins_end, 873
__gnu_parallel::_DRandomShufflingGlobalData< _RAIter >, 871
_DRandomShufflingGlobalData, 872
_M_bin_proc, 872
_M_dist, 872
_M_num_bins, 872
_M_num_bits, 872
_M_source, 873
_M_starts, 873
_M_temporaries, 873
__gnu_parallel::_DummyReduct, 874
__gnu_parallel::_EqualFromLess< _T1, _T2, _Compare >, 875
first_argument_type, 876
result_type, 876
second_argument_type, 876
__gnu_parallel::_EqualTo< _T1, _T2 >, 876

first_argument_type, 877
result_type, 877
second_argument_type, 877
__gnu_parallel::_GuardedIterator< _RAIter, _Compare >, 886
_GuardedIterator, 886
operator _RAIter, 887
operator<, 887
operator<=, 887
operator++, 887
operator*, 887
__gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >, 893
first, 899
first_type, 895, 896
operator<=>, 896, 897
operator==, 897, 898
second, 899
second_type, 896
swap, 896, 898
__gnu_parallel::_IteratorTriple< _Iterator1, _Iterator2, _Iterator3, _IteratorCategory >, 899
__gnu_parallel::_Job< _DifferenceTp >, 900
_M_first, 901
_M_last, 901
_M_load, 901
__gnu_parallel::_Less< _T1, _T2 >, 902
first_argument_type, 903
result_type, 903
second_argument_type, 903
__gnu_parallel::_Lexicographic< _T1, _T2, _Compare >, 903
first_argument_type, 904
result_type, 904
second_argument_type, 904
__gnu_parallel::_LexicographicReverse< _T1, _T2, _Compare >, 905
first_argument_type, 905
result_type, 905
second_argument_type, 906
__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >, 913
_M_log_k, 915
__delete_min_insert, 914
__get_min_source, 914
__insert_start, 914
__gnu_parallel::_LoserTree< false, _Tp, _Compare >, 915
_M_log_k, 917
__delete_min_insert, 916
__get_min_source, 916
__init_winner, 916
__insert_start, 917
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >, 917
_LoserTreeBase, 918
_M_comp, 919
_M_first_insert, 919
_M_log_k, 919
_M_losers, 919
__get_min_source, 919
__insert_start, 919
~_LoserTreeBase, 918
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser, 912
_M_key, 912
_M_source, 912
_M_sup, 912
__gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >, 920
__gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >, 921
__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >, 922
__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >::_Loser, 913
__gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >, 922
__gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >, 923
__gnu_parallel::_LoserTreePointerUnguardedBase< _Tp, _Compare >, 925
__gnu_parallel::_LoserTreeTraits< _Tp >, 925
_M_use_pointer, 926
__gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >, 926
__gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >, 927
__gnu_parallel::_LoserTreeUnguardedBase< _Tp, _Compare >, 928
__gnu_parallel::_Multiplies< _Tp1, _Tp2, _Result >, 930
first_argument_type, 931
result_type, 931
second_argument_type, 931
__gnu_parallel::_Nothing, 941
operator(), 941
__gnu_parallel::_PMWMSSortingData< _RAIter >, 944
_M_num_threads, 944
_M_offsets, 944
_M_pieces, 944
_M_samples, 944
_M_source, 945
_M_starts, 945
_M_temporary, 945
__gnu_parallel::_Piece< _DifferenceTp >, 941
_M_begin, 942
_M_end, 942
__gnu_parallel::_Plus< _Tp1, _Tp2, _Result >, 942
first_argument_type, 943

result_type, 943
second_argument_type, 943
__gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >, 947
_PseudoSequence, 948
begin, 948
end, 948
__gnu_parallel::_PseudoSequenceIterator< _Tp, _DifferenceTp >, 948
__gnu_parallel::_QSBThreadLocal< _RAIter >, 949
_M_elements_leftover, 950
_M_global, 950
_M_initial, 950
_M_leftover_parts, 950
_M_num_threads, 950
_Piece, 950
_QSBThreadLocal, 950
__gnu_parallel::_RandomNumber, 951
_RandomNumber, 951
__genrand_bits, 952
operator(), 952
__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >, 954
_RestrictedBoundedConcurrentQueue, 955
~_RestrictedBoundedConcurrentQueue, 955
pop_back, 955
pop_front, 955
push_front, 956
__gnu_parallel::_SamplingSorter< __stable, _RAIter, _StrictWeakOrdering >, 1006
__gnu_parallel::_SamplingSorter< false, _RAIter, _StrictWeakOrdering >, 1006
__gnu_parallel::_Settings, 1008
accumulate_minimal_n, 1010
adjacent_difference_minimal_n, 1010
cache_line_size, 1010
count_minimal_n, 1011
fill_minimal_n, 1011
find_increasing_factor, 1011
find_initial_block_size, 1011
find_maximum_block_size, 1011
find_scale_factor, 1011
find_sequential_search_size, 1011
for_each_minimal_n, 1011
generate_minimal_n, 1011
get, 1010
L1_cache_size, 1012
L2_cache_size, 1012
max_element_minimal_n, 1012
merge_minimal_n, 1012
merge_oversampling, 1012
min_element_minimal_n, 1012
multiway_merge_minimal_k, 1012
multiway_merge_minimal_n, 1012
multiway_merge_oversampling, 1012
nth_element_minimal_n, 1013
partial_sort_minimal_n, 1013
partial_sum_dilation, 1013
partial_sum_minimal_n, 1013
partition_chunk_share, 1013
partition_chunk_size, 1013
partition_minimal_n, 1013
qsb_steals, 1013
random_shuffle_minimal_n, 1013
replace_minimal_n, 1014
search_minimal_n, 1014
set, 1010
set_difference_minimal_n, 1014
set_intersection_minimal_n, 1014
set_symmetric_difference_minimal_n, 1014
set_union_minimal_n, 1014
sort_minimal_n, 1014
sort_mwms_oversampling, 1014
sort_qs_num_samples_preset, 1014
sort_qsb_base_case_maximal_n, 1015
TLB_size, 1015
transform_minimal_n, 1015
unique_copy_minimal_n, 1015
__gnu_parallel::_SplitConsistently< __exact, _RAIter, _Compare, _SortingPlacesIterator >, 1016
__gnu_parallel::_SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIterator >, 1016
__gnu_parallel::_SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator >, 1017
__gnu_parallel::__accumulate_binop_reduct< _BinOp >, 728
__gnu_parallel::__accumulate_selector< _It >, 729
_M_finish_iterator, 730
operator(), 730
__gnu_parallel::__adjacent_difference_selector< _It >, 730
_M_finish_iterator, 731
__gnu_parallel::__adjacent_find_selector, 731
_M_sequential_algorithm, 732
operator(), 732
__gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >, 740
argument_type, 741
result_type, 741
__gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >, 741
argument_type, 742
result_type, 742
__gnu_parallel::__count_if_selector< _It, _Diff >, 746
_M_finish_iterator, 747
operator(), 747

__gnu_parallel::__count_selector< _It, _Diff >, 747
_M_finish_iterator, 749
operator(), 748
__gnu_parallel::__fill_selector< _It >, 764
_M_finish_iterator, 765
operator(), 765
__gnu_parallel::__find_first_of_selector< _FIterator >, 765
_M_sequential_algorithm, 766
operator(), 766
__gnu_parallel::__find_if_selector, 767
_M_sequential_algorithm, 767
operator(), 768
__gnu_parallel::__for_each_selector< _It >, 768
_M_finish_iterator, 769
operator(), 769
__gnu_parallel::__generate_selector< _It >, 770
_M_finish_iterator, 771
operator(), 770
__gnu_parallel::__generic_find_selector, 771
__gnu_parallel::__generic_for_each_selector< _It >, 771
_M_finish_iterator, 773
__gnu_parallel::__identity_selector< _It >, 773
_M_finish_iterator, 774
operator(), 774
__gnu_parallel::__inner_product_selector< _It, _It2, _Tp >, 774
_M_finish_iterator, 776
__begin1_iterator, 776
__begin2_iterator, 776
__inner_product_selector, 775
operator(), 775
__gnu_parallel::__max_element_reduct< _Compare, _It >, 777
__gnu_parallel::__min_element_reduct< _Compare, _It >, 777
__gnu_parallel::__mismatch_selector, 778
_M_sequential_algorithm, 779
operator(), 779
__gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >, 782
__gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >, 782
__gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >, 783
__gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >, 783
__gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >, 783
__gnu_parallel::__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >, 784
__gnu_parallel::__replace_if_selector< _It, _Op, _Tp >, 800
_M_finish_iterator, 801
__new_val, 801
__replace_if_selector, 801
operator(), 801
__gnu_parallel::__replace_selector< _It, _Tp >, 801
_M_finish_iterator, 803
__new_val, 803
__replace_selector, 802
operator(), 803
__gnu_parallel::__transform1_selector< _It >, 804
_M_finish_iterator, 805
operator(), 804
__gnu_parallel::__transform2_selector< _It >, 805
_M_finish_iterator, 806
operator(), 805
__gnu_parallel::__unary_negate< _Predicate, argument_type >, 806
argument_type, 807
result_type, 807
__gnu_parallel::balanced_quicksort_tag, 1085
__get_num_threads, 1086
set_num_threads, 1086
__gnu_parallel::balanced_tag, 1087
__get_num_threads, 1087
set_num_threads, 1087
__gnu_parallel::constant_size_blocks_tag, 1886
__gnu_parallel::default_parallel_tag, 2023
__get_num_threads, 2023
set_num_threads, 2023
__gnu_parallel::equal_split_tag, 2107
__gnu_parallel::exact_tag, 2115
__get_num_threads, 2115
set_num_threads, 2115
__gnu_parallel::find_tag, 2131
__gnu_parallel::growing_blocks_tag, 2202
__gnu_parallel::multiway_mergesort_exact_tag, 2524
__get_num_threads, 2524
set_num_threads, 2524
__gnu_parallel::multiway_mergesort_sampling_tag, 2525
__get_num_threads, 2525
set_num_threads, 2525
__gnu_parallel::multiway_mergesort_tag, 2526
__get_num_threads, 2526
set_num_threads, 2526
__gnu_parallel::omp_loop_static_tag, 2682
__get_num_threads, 2683
set_num_threads, 2683
__gnu_parallel::omp_loop_tag, 2683
__get_num_threads, 2684

set_num_threads, 2684
__gnu_parallel::parallel_tag, 2718
__get_num_threads, 2720
parallel_tag, 2720
set_num_threads, 2720
__gnu_parallel::quicksort_tag, 2779
__get_num_threads, 2779
set_num_threads, 2779
__gnu_parallel::sampling_tag, 2858
__get_num_threads, 2858
set_num_threads, 2859
__gnu_parallel::sequential_tag, 2926
__gnu_parallel::unbalanced_tag, 3141
__get_num_threads, 3141
set_num_threads, 3141
__gnu_pbds, 462
__gnu_pbds::associative_tag, 1068
__gnu_pbds::basic_branch< Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc >, 1088
__gnu_pbds::basic_branch_tag, 1089
__gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc >, 1160
__gnu_pbds::basic_hash_tag, 1161
__gnu_pbds::basic_invalidation_guarantee, 1202
__gnu_pbds::binary_heap_tag, 1744
__gnu_pbds::binomial_heap_tag, 1757
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1776
cc_hash_max_collision_check_resize_trigger, 1777
external_load_access, 1777
get_load, 1777
is_grow_needed, 1777
is_resize_needed, 1777
notify_cleared, 1778
notify_erase_search_collision, 1778
notify_erase_search_end, 1778
notify_erase_search_start, 1778
notify_erased, 1778
notify_externally_resized, 1778
notify_find_search_collision, 1778
notify_find_search_end, 1779
notify_find_search_start, 1779
notify_insert_search_collision, 1779
notify_insert_search_end, 1779
notify_insert_search_start, 1779
notify_inserted, 1779
notify_resized, 1779
set_load, 1780
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >, 1780
cc_hash_table, 1781–1784
__gnu_pbds::cc_hash_tag, 1784
__gnu_pbds::container_error, 1896
what, 1897
__gnu_pbds::container_tag, 1897
__gnu_pbds::container_traits< Cntnr >, 1898
erase_can_throw, 1899
order_preserving, 1899
reverse_iteration, 1899
split_join_can_throw, 1899
__gnu_pbds::container_traits_base< _Tag >, 1899
__gnu_pbds::container_traits_base< binary_heap_tag >, 1899
__gnu_pbds::container_traits_base< binomial_heap_tag >, 1900
__gnu_pbds::container_traits_base< cc_hash_tag >, 1900
__gnu_pbds::container_traits_base< gp_hash_tag >, 1900
__gnu_pbds::container_traits_base< list_update_tag >, 1901
__gnu_pbds::container_traits_base< ov_tree_tag >, 1901
__gnu_pbds::container_traits_base< pairing_heap_tag >, 1901
__gnu_pbds::container_traits_base< pat_trie_tag >, 1901
__gnu_pbds::container_traits_base< rb_tree_tag >, 1902
__gnu_pbds::container_traits_base< rc_binomial_heap_tag >, 1902
__gnu_pbds::container_traits_base< splay_tree_tag >, 1902
__gnu_pbds::container_traits_base< thin_heap_tag >, 1903
__gnu_pbds::detail::bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >, 1720
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1722
const_reference, 1723
difference_type, 1723
get_l_child, 1724
get_metadata, 1724
get_r_child, 1725
iterator_category, 1724
metadata_const_reference, 1724
metadata_type, 1724
operator!=, 1725
operator==, 1725
operator*, 1725
reference, 1724
value_type, 1724
__gnu_pbds::detail::bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc

>, 1725
__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1727
const_reference, 1728
difference_type, 1728
get_l_child, 1729
get_metadata, 1729
get_r_child, 1729
iterator_category, 1728
metadata_const_reference, 1728
metadata_type, 1729
operator!=, 1729
operator==, 1730
operator*, 1730
reference, 1729
value_type, 1729
__gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >, 1730
node_const_iterator, 1731
__gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >, 1731
node_const_iterator, 1732
__gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >, 1735
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1737
binary_heap_const_iterator_, 1739
const_pointer, 1738
const_reference, 1738
difference_type, 1738
iterator_category, 1738
operator!=, 1739
operator->, 1740
operator==, 1740
operator*, 1740
pointer, 1739
reference, 1739
value_type, 1739
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1740
binary_heap_point_const_iterator_, 1743
const_pointer, 1742
const_reference, 1742
difference_type, 1742
iterator_category, 1742
operator!=, 1743
operator->, 1743
operator==, 1743
operator*, 1743
pointer, 1742
reference, 1742
value_type, 1742
__gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc >, 1752
__gnu_pbds::detail::binomial_heap_base< Value_Type, Cmp_Fn, _Alloc >, 1754
__gnu_pbds::detail::branch_policy< Node_CItr, Node_CItr, _Alloc >, 1773
__gnu_pbds::detail::branch_policy< Node_CItr, Node_Itr, _Alloc >, 1772
__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >, 1785
empty, 1788
get_comb_hash_fn, 1788
get_eq_fn, 1789
get_hash_fn, 1789
get_resize_policy, 1789
__gnu_pbds::detail::cond_dealtor< Entry, _Alloc >, 1876
__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, Tag, Policy_Tl >, 1887
__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, cc_hash_tag, Policy_Tl >, 1890
type, 1890
__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, gp_hash_tag, Policy_Tl >, 1890
type, 1891
__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, list_update_tag, Policy_Tl >, 1891
type, 1891
__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, ov_tree_tag, Policy_Tl >, 1891
type, 1892
__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, pat_trie_tag, Policy_Tl >, 1892
__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, rb_tree_tag, Policy_Tl >, 1892
type, 1892
__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, splay_tree_tag, Policy_Tl >, 1893
type, 1893
__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, cc_hash_tag, Policy_Tl >, 1893
type, 1893
__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, gp_hash_tag, Policy_Tl >, 1894
type, 1894
__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, list_update_tag, Policy_Tl >, 1894
type, 1894
__gnu_pbds::detail::container_base_dispatch< Key,

null_type, _Alloc, ov_tree_tag, Policy_TI >, 1895
type, 1895
__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, pat_trie_tag, Policy_TI >, 1895
type, 1895
__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, rb_tree_tag, Policy_TI >, 1896
__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, splay_tree_tag, Policy_TI >, 1896
type, 1896
__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binary_heap_tag, null_type >, 1887
type, 1888
__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binomial_heap_tag, null_type >, 1888
type, 1888
__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, pairing_heap_tag, null_type >, 1888
type, 1889
__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type >, 1889
type, 1889
__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, thin_heap_tag, null_type >, 1889
type, 1890
__gnu_pbds::detail::default_comb_hash_fn, 2019
type, 2019
__gnu_pbds::detail::default_eq_fn< Key >, 2022
type, 2022
__gnu_pbds::detail::default_hash_fn< Key >, 2022
type, 2022
__gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn >, 2024
type, 2024
__gnu_pbds::detail::default_resize_policy< Comb_Hash_Fn >, 2024
type, 2024
__gnu_pbds::detail::default_trie_access_traits< Key >, 2025
__gnu_pbds::detail::default_trie_access_traits< std::basic_string< Char, Char_Traits, std::allocator< char > > >, 2025
type, 2025
__gnu_pbds::detail::default_update_policy, 2026
type, 2026
__gnu_pbds::detail::dumnode_const_iterator< Key, Data, _Alloc >, 2067
__gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, No_Throw >, 2105
__gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, false >, 2105
__gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, false >::type, 3132
__gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, true >, 2105
type, 2106
__gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, No_Throw >, 2106
__gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, false >, 2106
__gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, true >, 2106
__gnu_pbds::detail::eq_by_less< Key, Cmp_Fn >, 2107
__gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >, 2191
empty, 2194
get_comb_probe_fn, 2194
get_eq_fn, 2195
get_hash_fn, 2195
get_probe_fn, 2195
get_resize_policy, 2195, 2196
__gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, Store_Hash >, 2221
__gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, false >, 2221
__gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, true >, 2222
__gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, Hold_Size >, 2226
__gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, true >, 2226
__gnu_pbds::detail::left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >, 2302
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >, 2304
const_pointer, 2305
const_reference, 2305
difference_type, 2305
iterator_category, 2306
left_child_next_sibling_heap_const_iterator_, 2306
operator!=, 2306, 2307
operator->, 2307
operator==, 2307
operator*, 2307
pointer, 2306
reference, 2306
value_type, 2306
__gnu_pbds::detail::left_child_next_sibling_heap_node_<

_Value, _Metadata, _Alloc >, 2307
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >, 2308
const_pointer, 2309
const_reference, 2309
difference_type, 2309
iterator_category, 2310
left_child_next_sibling_heap_node_point_const_iterator_, 2310
operator!=, 2311
operator->, 2311
operator==, 2311
operator*, 2311
pointer, 2310
reference, 2310
value_type, 2310
__gnu_pbds::detail::lu_counter_metadata< Size_Type >, 2369
__gnu_pbds::detail::lu_counter_policy_base< Size_Type >, 2371
__gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >, 2372
__gnu_pbds::detail::mask_based_range_hashing< Size_Type >, 2408
__gnu_pbds::detail::maybe_null_type< Key, Mapped, _Alloc, Store_Hash >, 2417
__gnu_pbds::detail::maybe_null_type< Key, null_type, _Alloc, Store_Hash >, 2419
__gnu_pbds::detail::mod_based_range_hashing< Size_Type >, 2437
__gnu_pbds::detail::no_throw_copies< Key, Mapped >, 2535
__gnu_pbds::detail::no_throw_copies< Key, null_type >, 2536
__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >, 2695
node_begin, 2697
node_end, 2697
__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::cond_dtor< Size_Type >, 1876
__gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >, 2698
get_l_child, 2699
get_r_child, 2699
__gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >, 2700
get_l_child, 2701
get_r_child, 2701
operator*, 2701
__gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc >, 2715
__gnu_pbds::detail::pat_trie_base, 2731
node_type, 2732
__gnu_pbds::detail::pat_trie_base::_CIter< Node, Leaf, Head, Inode, Is_Forward_Iterator >, 865
__gnu_pbds::detail::pat_trie_base::_Head< _ATraits, Metadata >, 888
__gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >, 889
__gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >::const_iterator, 1879
__gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >::iterator, 2296
__gnu_pbds::detail::pat_trie_base::_Iter< Node, Leaf, Head, Inode, Is_Forward_Iterator >, 891
__gnu_pbds::detail::pat_trie_base::_Leaf< _ATraits, Metadata >, 901
__gnu_pbds::detail::pat_trie_base::_Metadata< Metadata, _Alloc >, 929
__gnu_pbds::detail::pat_trie_base::_Metadata< null_type, _Alloc >, 929
__gnu_pbds::detail::pat_trie_base::_Node_base< _ATraits, Metadata >, 933
__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 934
get_child, 936
get_metadata, 936
metadata_const_reference, 935
metadata_type, 935
num_children, 936
operator!=, 936
operator==, 936
operator*, 936
valid_prefix, 937
__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 937
get_child, 938
get_metadata, 939
metadata_const_reference, 938
metadata_type, 938
num_children, 939
operator!=, 939
operator==, 939
operator*, 939
valid_prefix, 939
__gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >, 2732
node_begin, 2734
node_end, 2735
node_type, 2734
__gnu_pbds::detail::probe_fn_base< _Alloc >, 2770
__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash >, 2784
__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, false >, 2784

__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true >, 2785
__gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, false >, 2785
__gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, true >, 2786
__gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_Hash >, 2787
__gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, false >, 2787
__gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true >, 2788
__gnu_pbds::detail::ranged_probe_fn< Key, null_type, _Alloc, Comb_Probe_Fn, null_type, false >, 2789
__gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >, 2801
  node_begin, 2804
  node_end, 2804
__gnu_pbds::detail::rb_tree_node_< Value_Type, Metadata, _Alloc >, 2804
__gnu_pbds::detail::rc< _Node, _Alloc >, 2806
__gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc >, 2807
__gnu_pbds::detail::rebind_traits< _Alloc, T >, 2809
__gnu_pbds::detail::resize_policy< _Tp >, 2833
__gnu_pbds::detail::select_value_type< Key, Mapped >, 2863
__gnu_pbds::detail::select_value_type< Key, null_type >, 2863
__gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >, 2980
  node_begin, 2983
  node_end, 2983
__gnu_pbds::detail::splay_tree_node_< Value_Type, Metadata, _Alloc >, 2983
__gnu_pbds::detail::stored_data< _Tv, _Th, Store_Hash >, 3027
__gnu_pbds::detail::stored_data< _Tv, _Th, false >, 3028
__gnu_pbds::detail::stored_hash< _Th >, 3028
__gnu_pbds::detail::stored_value< _Tv >, 3029
__gnu_pbds::detail::synth_access_traits< Type_Traits, Set, _ATraits >, 3044
__gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc >, 3049
__gnu_pbds::detail::tree_metadata_helper< Node_Update, _BTp >, 3096
__gnu_pbds::detail::tree_metadata_helper< Node_Update, false >, 3096
__gnu_pbds::detail::tree_metadata_helper< Node_Update, true >, 3096
__gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >, 3097
__gnu_pbds::detail::tree_traits< Key, Data, Cmp_Fn, Node_Update, Tag, _Alloc >, 3099
__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >, 3100
  node_const_iterator, 3100
__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >, 3100
  node_const_iterator, 3101
__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >, 3101
  node_const_iterator, 3103
__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >, 3103
  node_const_iterator, 3103
__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >, 3103
  node_const_iterator, 3105
__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >, 3105
  node_const_iterator, 3106
__gnu_pbds::detail::trie_metadata_helper< Node_Update, _BTp >, 3109
__gnu_pbds::detail::trie_metadata_helper< Node_Update, false >, 3109
__gnu_pbds::detail::trie_metadata_helper< Node_Update, true >, 3109
__gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >, 3110
__gnu_pbds::detail::trie_policy_base< Node_CItr, Node_Itr, _ATraits, _Alloc >, 3113
  end, 3114
__gnu_pbds::detail::trie_traits< Key, Data, _ATraits, Node_Update, Tag, _Alloc >, 3120
__gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >, 3121
  node_const_iterator, 3121
  node_update, 3121
  synth_access_traits, 3121
__gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >, 3122
  node_const_iterator, 3122
  node_update, 3122
  synth_access_traits, 3123
__gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash >, 3134
__gnu_pbds::direct_mask_range_hashing< Size_Type >, 2051
  operator(), 2052
__gnu_pbds::direct_mod_range_hashing< Size_Type >, 2053
  operator(), 2053
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn,

Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >, 2185
gp_hash_table, 2186–2190
__gnu_pbds::gp_hash_tag, 2190
__gnu_pbds::hash_exponential_size_policy< Size_Type >, 2222
hash_exponential_size_policy, 2223
__gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >, 2223
external_load_access, 2225
get_loads, 2225
hash_load_check_resize_trigger, 2225
notify_cleared, 2225
notify_inserted, 2225
notify_resized, 2225
set_loads, 2226
__gnu_pbds::hash_prime_size_policy, 2232
hash_prime_size_policy, 2232
size_type, 2232
__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >, 2234
external_load_access, 2235
get_actual_size, 2236
get_loads, 2236
get_new_size, 2236
get_size_policy, 2237
get_trigger_policy, 2237
hash_standard_resize_policy, 2236
resize, 2237
set_loads, 2237
__gnu_pbds::insert_error, 2250
what, 2251
__gnu_pbds::join_error, 2301
what, 2302
__gnu_pbds::linear_probe_fn< Size_Type >, 2323
operator(), 2324
__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >, 2347
list_update, 2348
__gnu_pbds::list_update_tag, 2348
__gnu_pbds::lu_counter_policy< Max_Count, _Alloc >, 2369
max_count, 2371
metadata_reference, 2370
metadata_type, 2370
operator(), 2371
__gnu_pbds::lu_move_to_front_policy< _Alloc >, 2375
metadata_reference, 2375
metadata_type, 2375
operator(), 2375
__gnu_pbds::null_node_update< _Tp1, _Tp2, _Tp3, _Tp4 >, 2542
__gnu_pbds::null_type, 2543
__gnu_pbds::ov_tree_tag, 2701
__gnu_pbds::pairing_heap_tag, 2718
__gnu_pbds::pat_trie_tag, 2735
__gnu_pbds::point_invalidation_guarantee, 2752
__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >, 2764
priority_queue, 2765
__gnu_pbds::priority_queue_tag, 2769
__gnu_pbds::quadratic_probe_fn< Size_Type >, 2775
operator(), 2775
__gnu_pbds::range_invalidation_guarantee, 2783
__gnu_pbds::rb_tree_tag, 2805
__gnu_pbds::rc_binomial_heap_tag, 2809
__gnu_pbds::resize_error, 2832
what, 2832
__gnu_pbds::sample_probe_fn, 2844
operator(), 2845
sample_probe_fn, 2845
swap, 2845
__gnu_pbds::sample_range_hashing, 2845
notify_resized, 2846
operator(), 2846
sample_range_hashing, 2846
size_type, 2846
swap, 2846
__gnu_pbds::sample_ranged_hash_fn, 2846
notify_resized, 2847
operator(), 2847
sample_ranged_hash_fn, 2847
swap, 2847
__gnu_pbds::sample_ranged_probe_fn, 2847
__gnu_pbds::sample_resize_policy, 2848
get_new_size, 2849
is_resize_needed, 2849
notify_cleared, 2849
notify_erase_search_collision, 2849
notify_erase_search_end, 2849
notify_erase_search_start, 2849
notify_erased, 2849
notify_find_search_collision, 2849
notify_find_search_end, 2850
notify_find_search_start, 2850
notify_insert_search_collision, 2850
notify_insert_search_end, 2850
notify_insert_search_start, 2850
notify_inserted, 2850
notify_resized, 2850
sample_range_hashing, 2850
sample_resize_policy, 2849
size_type, 2849
swap, 2850
__gnu_pbds::sample_resize_trigger, 2851
is_grow_needed, 2852
is_resize_needed, 2852

notify_cleared, 2852
notify_erase_search_collision, 2852
notify_erase_search_end, 2852
notify_erase_search_start, 2852
notify_erased, 2852
notify_externally_resized, 2852
notify_find_search_collision, 2852
notify_find_search_end, 2852
notify_find_search_start, 2852
notify_insert_search_collision, 2853
notify_insert_search_end, 2853
notify_insert_search_start, 2853
notify_inserted, 2853
notify_resized, 2853
sample_range_hashing, 2853
sample_resize_trigger, 2851
size_type, 2851
swap, 2853
__gnu_pbds::sample_size_policy, 2853
get_nearest_larger_size, 2854
get_nearest_smaller_size, 2854
sample_range_hashing, 2854
sample_size_policy, 2854
size_type, 2854
swap, 2854
__gnu_pbds::sample_tree_node_update< Const_Node_Iter, Node_Iter, Cmp_Fn, _Alloc >, 2855
__gnu_pbds::sample_trie_access_traits, 2855
begin, 2855
e_pos, 2855
e_type, 2855
end, 2856
__gnu_pbds::sample_trie_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >, 2856
operator(), 2856
sample_trie_node_update, 2856
__gnu_pbds::sample_update_policy, 2857
metadata_type, 2857
operator(), 2857
sample_update_policy, 2857
swap, 2858
__gnu_pbds::sequence_tag, 2925
__gnu_pbds::splay_tree_tag, 2984
__gnu_pbds::string_tag, 3030
__gnu_pbds::thin_heap_tag, 3051
__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >, 3093
cmp_fn, 3095
tree, 3095
__gnu_pbds::tree_order_statistics_node_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc >, 3097
find_by_order, 3098
operator(), 3098
order_of_key, 3098
__gnu_pbds::tree_tag, 3099
__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >, 3106
access_traits, 3108
trie, 3108
__gnu_pbds::trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >, 3110
find_by_order, 3112
operator(), 3112
order_of_key, 3112
order_of_prefix, 3112
__gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >, 3114
a_const_iterator, 3116
access_traits, 3116
allocator_type, 3116
operator(), 3116
prefix_range, 3116, 3117
size_type, 3116
__gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >, 3117
begin, 3118
const_iterator, 3118
e_pos, 3119
e_type, 3118
end, 3119
__gnu_pbds::trie_tag, 3119
__gnu_pbds::trivial_iterator_tag, 3123
__gnu_sequential, 464
__init_winner
__gnu_parallel::_LoserTree< false, _Tp, _Compare >, 916
__inner_product_selector
__gnu_parallel::__inner_product_selector< _It, _It2, _Tp >, 775
__inplace_stable_sort
std, 582
__insert_start
__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >, 914
__gnu_parallel::_LoserTree< false, _Tp, _Compare >, 917
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >, 919
__int_traits
__gnu_cxx, 394
__invoke
Utilities, 299
__is_sorted
__gnu_parallel, 432
__iterator_category
Iterators, 170
__lg
std, 582

__median
SGI, 150
__median_of_three_iterators
__gnu_parallel, 432
__merge_adaptive
std, 582
__merge_advance
__gnu_parallel, 433
__merge_advance_movc
__gnu_parallel, 433
__merge_advance_usual
__gnu_parallel, 434
__merge_without_buffer
std, 583
__move_median_to_first
std, 583
__move_merge
std, 583
__move_merge_adaptive
std, 583
__move_merge_adaptive_backward
std, 584
__multiline
std::regex_constants, 715
__new_val
__gnu_parallel::__replace_if_selector< _It, _Op, _Tp >, 801
__gnu_parallel::__replace_selector< _It, _Tp >, 803
__num_bitmaps
__gnu_cxx::__detail, 404
__num_blocks
__gnu_cxx::__detail, 405
__num_get_type
std::basic_ios< _CharT, _Traits >, 1207
std::basic_ofstream< _CharT, _Traits >, 1353
std::basic_ostream< _CharT, _Traits >, 1387
std::basic_ostream< _CharT, _Traits >::sentry, 2902
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1419
__num_put_type
std::basic_ifstream< _CharT, _Traits >, 1168
std::basic_ios< _CharT, _Traits >, 1207
std::basic_istream< _CharT, _Traits >, 1277
std::basic_istream< _CharT, _Traits >::sentry, 2869
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1314
__parallel_merge_advance
__gnu_parallel, 435
__parallel_nth_element
__gnu_parallel, 436
__parallel_partial_sort
__gnu_parallel, 436
__parallel_partial_sum
__gnu_parallel, 437
__parallel_partial_sum_basecase
__gnu_parallel, 437
__parallel_partial_sum_linear
__gnu_parallel, 437
__parallel_partition
__gnu_parallel, 438
__parallel_random_shuffle
__gnu_parallel, 438
__parallel_random_shuffle_drs
__gnu_parallel, 439
__parallel_random_shuffle_drs_pu
__gnu_parallel, 439
__parallel_sort
__gnu_parallel, 440–444
__parallel_sort_qs
__gnu_parallel, 445
__parallel_sort_qs_conquer
__gnu_parallel, 445
__parallel_sort_qs_divide
__gnu_parallel, 445
__parallel_sort_qsb
__gnu_parallel, 446
__parallel_unique_copy
__gnu_parallel, 446, 447
__partition
std, 584
__polynomial
std::regex_constants, 715
__ptr_rebind
std, 578
__qsb_conquer
__gnu_parallel, 447
__qsb_divide
__gnu_parallel, 448
__qsb_local_sort_with_helping
__gnu_parallel, 448
__random_number_pow2
__gnu_parallel, 449
__rd_log2
__gnu_parallel, 449
__replace_if_selector
__gnu_parallel::__replace_if_selector< _It, _Op, _Tp >, 801
__replace_selector
__gnu_parallel::__replace_selector< _It, _Tp >, 802
__resize_and_overwrite
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1481
std::basic_string< _CharT, _Traits, _Alloc >, 1561
__reverse
std, 584
__rotate
std, 585
__rotate_adaptive

std, 585
__round_up_to_pow2
__gnu_parallel, 449
__sample
std, 585, 586
__search_n_aux
std, 586
__search_template
__gnu_parallel, 449
__sequential_multiway_merge
__gnu_parallel, 450
__sequential_random_shuffle
__gnu_parallel, 450
__shrink
__gnu_parallel, 451
__shrink_and_double
__gnu_parallel, 451
__stable_partition_adaptive
std, 586
__static_pointer_cast
__gnu_cxx, 394
__streambuf_type
std::basic_streambuf< _CharT, _Traits >, 1459
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3331
__umap_traits
std, 578
__ummap_traits
std, 579
__umset_traits
std, 579
__uset_traits
std, 579
__valid_range
__gnu_debug, 413
__valid_range_aux
__gnu_debug, 414
__verbose_terminate_handler
Exceptions, 129
__versa_string
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 811–813
__yield
__gnu_parallel, 452
~_LoserTreeBase
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >, 918
~_RestrictedBoundedConcurrentQueue
__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >, 955
~_Safe_sequence_base
__gnu_debug::_Safe_sequence_base, 996
~_Safe_unordered_container_base
__gnu_debug::_Safe_unordered_container_base, 1003
~__versa_string
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 814
~any
std::experimental::fundamentals_v1::any, 1065
~auto_ptr
std::auto_ptr< _Tp >, 1073
~basic_filebuf
std::basic_filebuf< _CharT, _Traits >, 1093
~basic_fstream
std::basic_fstream< _CharT, _Traits >, 1119
~basic_ifstream
std::basic_ifstream< _CharT, _Traits >, 1170
~basic_ios
std::basic_ios< _CharT, _Traits >, 1209
~basic_iostream
std::basic_iostream< _CharT, _Traits >, 1232
~basic_istream
std::basic_istream< _CharT, _Traits >, 1278
std::basic_istream< _CharT, _Traits >::sentry, 2871
~basic_istringstream
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1317
~basic_ofstream
std::basic_ofstream< _CharT, _Traits >, 1355
~basic_ostream
std::basic_ostream< _CharT, _Traits >, 1389
std::basic_ostream< _CharT, _Traits >::sentry, 2904
~basic_ostringstream
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1421
~basic_regex
std::basic_regex< _Ch_type, _Rx_traits >, 1450
~basic_streambuf
std::basic_streambuf< _CharT, _Traits >, 1460
~basic_string
std::basic_string< _CharT, _Traits, _Alloc >, 1557, 1560
~basic_stringstream
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1677
~collate
std::__cxx11::collate< _CharT >, 1858
~ctype
std::ctype< char >, 1927
std::ctype< wchar_t >, 1961
~deque
std::deque< _Tp, _Alloc >, 2036
~facet
std::locale::facet, 2127
~forward_list
std::forward_list< _Tp, _Alloc >, 2145
~gslice
Numeric Arrays, 221

~ios_base
std::ios_base, 2261
~list
std::list< _Tp, _Alloc >, 2332
~locale
std::locale, 2353
~map
std::map< _Key, _Tp, _Compare, _Alloc >, 2386
~match_results
std::match_results< _Bi_iter, _Alloc >, 2412
~messages
std::messages< _CharT >, 2431
~money_get
std::money_get< _CharT, _InIter >, 2442
~money_put
std::money_put< _CharT, _OutIter >, 2447
~moneypunct
std::moneypunct< _CharT, _Intl >, 2452
~multimap
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2482
~multiset
std::multiset< _Key, _Compare, _Alloc >, 2509
~num_get
std::num_get< _CharT, _InIter >, 2547
~num_put
std::num_put< _CharT, _OutIter >, 2563
~numpunct
std::numpunct< _CharT >, 2675
~sentry
std::basic_ostream< _CharT, _Traits >::sentry, 2904
~set
std::set< _Key, _Compare, _Alloc >, 2937
~stdio_filebuf
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2992
~temporary_buffer
__gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >, 3048
~time_get
std::time_get< _CharT, _InIter >, 3064
~time_put
std::time_put< _CharT, _OutIter >, 3086
~type_info
__gnu_debug::type_info, 3133
std::type_info, 3134
~unique_ptr
std::unique_ptr< _Tp, _Dp >, 3152
std::unique_ptr< _Tp[], _Dp >, 3158, 3160
~vector
std::vector< _Tp, _Alloc >, 3295
std::vector< bool, _Alloc >, 3316

a
std::extreme_value_distribution< _RealType >, 2123
std::weibull_distribution< _RealType >, 3347
a_const_iterator
__gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >, 3116
abi, 464
abs
Complex Numbers, 181
math.h, 3701
stdlib.h, 3757
Time, 366
access_traits
__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >, 3108
__gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >, 3116
accumulate
Generalized Numeric operations, 6, 7
accumulate_minimal_n
__gnu_parallel::_Settings, 1010
acosh
std, 587
Adaptors for pointers to functions, 309
ptr_fun, 309
Adaptors for pointers to members, 310
add_lvalue_reference_t
Metaprogramming, 350
add_pointer_t
Metaprogramming, 350
add_rvalue_reference_t
Metaprogramming, 350
addressof
Utilities, 299
adjacent_difference
Generalized Numeric operations, 7, 8
adjacent_difference_minimal_n
__gnu_parallel::_Settings, 1010
adjacent_find
Non-Mutating, 44
adjustfield
std::basic_fstream< _CharT, _Traits >, 1155
std::basic_ifstream< _CharT, _Traits >, 1197
std::basic_ios< _CharT, _Traits >, 1220
std::basic_iostream< _CharT, _Traits >, 1267
std::basic_istream< _CharT, _Traits >, 1304
std::basic_istream< _CharT, _Traits >::sentry, 2894
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1343
std::basic_ofstream< _CharT, _Traits >, 1378
std::basic_ostream< _CharT, _Traits >, 1408
std::basic_ostream< _CharT, _Traits >::sentry, 2922
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1442
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1712

std::ios_base, 2266
adopt_lock
Mutexes, 118
advance
std, 587
airy_ai
Mathematical Special Functions, 192
airy_aif
Mathematical Special Functions, 192
airy_ail
Mathematical Special Functions, 192
airy_bi
Mathematical Special Functions, 192
airy_bif
Mathematical Special Functions, 192
airy_bil
Mathematical Special Functions, 193
algo.h, 3714
algobase.h, 3723
algorithm, 3352–3354
algorithmfwd.h, 3364, 3369
Algorithms, 4
align
Memory, 318
align.h, 3377
aligned_buffer.h, 3605
aligned_storage_t
Metaprogramming, 350
alignment_value
Metaprogramming, 353
all
std::bitset< _Nb >, 1765
std::locale, 2357
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2074
all_of
Non-Mutating, 44
alloc_traits.h, 3377, 3378
allocate
__gnu_cxx::__alloc_traits< _Alloc, typename >, 735, 736
std::allocator_traits< _Alloc >, 1034
std::allocator_traits< allocator< _Tp > >, 1041
std::allocator_traits< allocator< void > >, 1049, 1051
std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1058, 1059
allocate_shared
Pointer Abstractions, 328
allocated_ptr.h, 3378
allocator.h, 3378
allocator_type
__gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >, 3116
std::allocator_traits< _Alloc >, 1032
std::allocator_traits< allocator< _Tp > >, 1038
std::allocator_traits< allocator< void > >, 1047
std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1055
std::set< _Key, _Compare, _Alloc >, 2932
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3171
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3201
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3230
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3256
Allocators, 323
__allocator_base, 324
operator==, 324
alpha
std::gamma_distribution< _RealType >, 2180
any, 3354, 3355
std::bitset< _Nb >, 1765
std::experimental::fundamentals_v1::any, 1065
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2074
any_cast
Type-safe container of any type, 288–290
any_of
Non-Mutating, 45
app
std::basic_fstream< _CharT, _Traits >, 1155
std::basic_ifstream< _CharT, _Traits >, 1198
std::basic_ios< _CharT, _Traits >, 1220
std::basic_iostream< _CharT, _Traits >, 1267
std::basic_istream< _CharT, _Traits >, 1304
std::basic_istream< _CharT, _Traits >::sentry, 2894
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1344
std::basic_ofstream< _CharT, _Traits >, 1378
std::basic_ostream< _CharT, _Traits >, 1408
std::basic_ostream< _CharT, _Traits >::sentry, 2922
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1442
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1712
std::ios_base, 2266
append
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 814–816
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1481, 1482
std::basic_string< _CharT, _Traits, _Alloc >, 1561–1566
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2074, 2075
apply
Numeric Arrays, 221, 222

apply_generator
__gnu_cxx::typelist, 405
arg
Complex Numbers, 181
std::tr1, 724
argument_type
__gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >, 1733
__gnu_cxx::select1st< _Pair >, 2862
__gnu_cxx::select2nd< _Pair >, 2862
__gnu_cxx::subtractive_rng, 3044
__gnu_cxx::unary_compose< _Operation1, _Operation2 >, 3138
__gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >, 741
__gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >, 742
__gnu_parallel::__unary_negate< _Predicate, argument_type >, 807
std::binder1st< _Operation >, 1746
std::binder2nd< _Operation >, 1748
std::const_mem_fun_ref_t< _Ret, _Tp >, 1884
std::const_mem_fun_t< _Ret, _Tp >, 1885
std::hash< __gnu_cxx::throw_value_limit >, 2209
std::hash< __gnu_cxx::throw_value_random >, 2210
std::logical_not< _Tp >, 2362
std::logical_not< void >, 2363
std::mem_fun_ref_t< _Ret, _Tp >, 2422
std::mem_fun_t< _Ret, _Tp >, 2423
std::negate< _Tp >, 2528
std::negate< void >, 2529
std::pointer_to_unary_function< _Arg, _Result >, 2755
std::unary_function< _Arg, _Result >, 3139
std::unary_negate< _Predicate >, 3140
Arithmetic Function Object Classes, 311
array, 3355, 3356
Array creation functions, 281
make_array, 282
to_array, 282
assertions.h, 3576
assign
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 816–819
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1483, 1484
std::basic_regex< _Ch_type, _Rx_traits >, 1451–1453
std::basic_string< _CharT, _Traits, _Alloc >, 1566–1572
std::deque< _Tp, _Alloc >, 2039, 2040
std::error_condition, 2114
std::forward_list< _Tp, _Alloc >, 2145, 2146
std::list< _Tp, _Alloc >, 2334
std::vector< _Tp, _Alloc >, 3295, 3296
std::vector< bool, _Alloc >, 3316, 3318
assoc_container.hpp, 3611
assoc_laguerre
Mathematical Special Functions, 193
TR1 Mathematical Special Functions, 248
assoc_laguerref
Mathematical Special Functions, 194
assoc_laguerrel
Mathematical Special Functions, 194
assoc_legendre
Mathematical Special Functions, 194
TR1 Mathematical Special Functions, 248
assoc_legendref
Mathematical Special Functions, 195
assoc_legendrel
Mathematical Special Functions, 195
Associative, 121
assume_aligned
Memory, 318
async
Futures, 114
at
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 819, 820
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1485–1487
std::basic_string< _CharT, _Traits, _Alloc >, 1572, 1573
std::deque< _Tp, _Alloc >, 2040, 2041
std::map< _Key, _Tp, _Compare, _Alloc >, 2387
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3175
std::vector< _Tp, _Alloc >, 3296, 3297
std::vector< bool, _Alloc >, 3318
ate
std::basic_fstream< _CharT, _Traits >, 1155
std::basic_ifstream< _CharT, _Traits >, 1198
std::basic_ios< _CharT, _Traits >, 1220
std::basic_iostream< _CharT, _Traits >, 1267
std::basic_istream< _CharT, _Traits >, 1304
std::basic_istream< _CharT, _Traits >::sentry, 2894
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1344
std::basic_ofstream< _CharT, _Traits >, 1378
std::basic_ostream< _CharT, _Traits >, 1408
std::basic_ostream< _CharT, _Traits >::sentry, 2922
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1442
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1712

std::ios_base, 2266
atomic, 3357
atomic_base.h, 3378
atomic_bool
Atomics, 104
ATOMIC_BOOL_LOCK_FREE
Atomics, 103
atomic_char
Atomics, 104
atomic_char16_t
Atomics, 104
atomic_char32_t
Atomics, 104
atomic_compare_exchange_strong
Pointer Abstractions, 329
atomic_compare_exchange_strong_explicit
Pointer Abstractions, 330
atomic_compare_exchange_weak
Pointer Abstractions, 331
atomic_compare_exchange_weak_explicit
Pointer Abstractions, 332
atomic_exchange
Pointer Abstractions, 333
atomic_exchange_explicit
Pointer Abstractions, 333, 334
atomic_futex.h, 3379
atomic_int
Atomics, 104
atomic_int16_t
Atomics, 104
atomic_int32_t
Atomics, 104
atomic_int64_t
Atomics, 104
atomic_int8_t
Atomics, 105
atomic_int_fast16_t
Atomics, 105
atomic_int_fast32_t
Atomics, 105
atomic_int_fast64_t
Atomics, 105
atomic_int_fast8_t
Atomics, 105
atomic_int_least16_t
Atomics, 105
atomic_int_least32_t
Atomics, 105
atomic_int_least64_t
Atomics, 106
atomic_int_least8_t
Atomics, 106
atomic_intmax_t
Atomics, 106
atomic_intptr_t
Atomics, 106
atomic_is_lock_free
Pointer Abstractions, 334
atomic_llong
Atomics, 106
atomic_load
Pointer Abstractions, 335
atomic_load_explicit
Pointer Abstractions, 335, 336
atomic_lockfree_defines.h, 3379
atomic_long
Atomics, 106
atomic_ptrdiff_t
Atomics, 106
atomic_schar
Atomics, 107
atomic_short
Atomics, 107
atomic_size_t
Atomics, 107
atomic_store
Pointer Abstractions, 336
atomic_store_explicit
Pointer Abstractions, 337
atomic_timed_wait.h, 3380
atomic_uchar
Atomics, 107
atomic_uint
Atomics, 107
atomic_uint16_t
Atomics, 107
atomic_uint32_t
Atomics, 107
atomic_uint64_t
Atomics, 108
atomic_uint8_t
Atomics, 108
atomic_uint_fast16_t
Atomics, 108
atomic_uint_fast32_t
Atomics, 108
atomic_uint_fast64_t
Atomics, 108
atomic_uint_fast8_t
Atomics, 108
atomic_uint_least16_t
Atomics, 108
atomic_uint_least32_t
Atomics, 109
atomic_uint_least64_t
Atomics, 109
atomic_uint_least8_t
Atomics, 109

atomic_uintmax_t
Atomics, 109
atomic_uintptr_t
Atomics, 109
atomic_ullong
Atomics, 109
atomic_ulong
Atomics, 109
atomic_ushort
Atomics, 110
atomic_wait.h, 3380
atomic_wchar_t
Atomics, 110
atomic_word.h, 3789
atomicity.h, 3606
Atomics, 99
atomic_bool, 104
ATOMIC_BOOL_LOCK_FREE, 103
atomic_char, 104
atomic_char16_t, 104
atomic_char32_t, 104
atomic_int, 104
atomic_int16_t, 104
atomic_int32_t, 104
atomic_int64_t, 104
atomic_int8_t, 105
atomic_int_fast16_t, 105
atomic_int_fast32_t, 105
atomic_int_fast64_t, 105
atomic_int_fast8_t, 105
atomic_int_least16_t, 105
atomic_int_least32_t, 105
atomic_int_least64_t, 106
atomic_int_least8_t, 106
atomic_intmax_t, 106
atomic_intptr_t, 106
atomic_llong, 106
atomic_long, 106
atomic_ptrdiff_t, 106
atomic_schar, 107
atomic_short, 107
atomic_size_t, 107
atomic_uchar, 107
atomic_uint, 107
atomic_uint16_t, 107
atomic_uint32_t, 107
atomic_uint64_t, 108
atomic_uint8_t, 108
atomic_uint_fast16_t, 108
atomic_uint_fast32_t, 108
atomic_uint_fast64_t, 108
atomic_uint_fast8_t, 108
atomic_uint_least16_t, 108
atomic_uint_least32_t, 109
atomic_uint_least64_t, 109
atomic_uint_least8_t, 109
atomic_uintmax_t, 109
atomic_uintptr_t, 109
atomic_ullong, 109
atomic_ulong, 109
atomic_ushort, 110
atomic_wchar_t, 110
kill_dependency, 110
memory_order, 110
auto_ptr
std::auto_ptr< _Tp >, 1073
auto_ptr.h, 3361
awk
std::regex_constants, 715

b
std::extreme_value_distribution< _RealType >, 2123
std::weibull_distribution< _RealType >, 3347
back
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 820
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1488
std::basic_string< _CharT, _Traits, _Alloc >, 1574
std::deque< _Tp, _Alloc >, 2041
std::list< _Tp, _Alloc >, 2335
std::queue< _Tp, _Sequence >, 2777
std::vector< _Tp, _Alloc >, 3297
std::vector< bool, _Alloc >, 3319
back_insert_iterator
std::back_insert_iterator< _Container >, 1078
back_inserter
Iterators, 171
backward_warning.h, 3361
bad
std::basic_fstream< _CharT, _Traits >, 1120
std::basic_ifstream< _CharT, _Traits >, 1171
std::basic_ios< _CharT, _Traits >, 1209
std::basic_iostream< _CharT, _Traits >, 1232
std::basic_istream< _CharT, _Traits >, 1279
std::basic_istream< _CharT, _Traits >::sentry, 2871
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1318
std::basic_ofstream< _CharT, _Traits >, 1356
std::basic_ostream< _CharT, _Traits >, 1389
std::basic_ostream< _CharT, _Traits >::sentry, 2904
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1421
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1678
badbit
std::basic_fstream< _CharT, _Traits >, 1155
std::basic_ifstream< _CharT, _Traits >, 1198

std::basic_ios< _CharT, _Traits >, 1221
std::basic_iostream< _CharT, _Traits >, 1267
std::basic_istream< _CharT, _Traits >, 1305
std::basic_istream< _CharT, _Traits >::sentry, 2894
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1344
std::basic_ofstream< _CharT, _Traits >, 1378
std::basic_ostream< _CharT, _Traits >, 1408
std::basic_ostream< _CharT, _Traits >::sentry, 2922
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1442
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1712
std::ios_base, 2266
balanced_quicksort.h, 3725
barrier, 3363
base
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 967
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 981
std::discard_block_engine< _RandomNumberEngine, __p, __r >, 2058
std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >, 2243
std::reverse_iterator< _Iterator >, 2836
std::shuffle_order_engine< _RandomNumberEngine, __k >, 2971
Base and Implementation Classes, 272
_Opcode, 272
Base and Policy Classes, 140–142
base.h, 3726
basefield
std::basic_fstream< _CharT, _Traits >, 1156
std::basic_ifstream< _CharT, _Traits >, 1198
std::basic_ios< _CharT, _Traits >, 1221
std::basic_iostream< _CharT, _Traits >, 1267
std::basic_istream< _CharT, _Traits >, 1305
std::basic_istream< _CharT, _Traits >::sentry, 2894
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1344
std::basic_ofstream< _CharT, _Traits >, 1378
std::basic_ostream< _CharT, _Traits >, 1408
std::basic_ostream< _CharT, _Traits >::sentry, 2922
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1442
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1713
std::ios_base, 2266
basic
std::regex_constants, 715
basic_file.h, 3789
basic_filebuf
std::basic_filebuf< _CharT, _Traits >, 1093
basic_fstream
std::basic_fstream< _CharT, _Traits >, 1118, 1119
basic_ifstream
std::basic_ifstream< _CharT, _Traits >, 1169, 1170
basic_ios
std::basic_ios< _CharT, _Traits >, 1209
basic_ios.h, 3380
basic_ios.tcc, 3380
basic_iostream
std::basic_iostream< _CharT, _Traits >, 1232
basic_istream
std::basic_istream< _CharT, _Traits >, 1278
std::basic_istream< _CharT, _Traits >::sentry, 2871
basic_istringstream
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1317
basic_iterator.h, 3727
basic_ofstream
std::basic_ofstream< _CharT, _Traits >, 1354, 1355
basic_ostream
std::basic_ostream< _CharT, _Traits >, 1389
std::basic_ostream< _CharT, _Traits >::sentry, 2904
basic_ostringstream
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1420
basic_regex
std::basic_regex< _Ch_type, _Rx_traits >, 1448–1450
basic_streambuf
std::basic_streambuf< _CharT, _Traits >, 1460
basic_string
std::basic_string< _CharT, _Traits, _Alloc >, 1554–1560
basic_string.h, 3381
basic_string.tcc, 3383
basic_stringbuf
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1653
basic_stringstream
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1677
before
__gnu_debug::type_info, 3133
std::type_info, 3134
before_begin
std::forward_list< _Tp, _Alloc >, 2146
beg
std::basic_fstream< _CharT, _Traits >, 1156
std::basic_ifstream< _CharT, _Traits >, 1198
std::basic_ios< _CharT, _Traits >, 1221
std::basic_iostream< _CharT, _Traits >, 1267
std::basic_istream< _CharT, _Traits >, 1305
std::basic_istream< _CharT, _Traits >::sentry, 2894
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1344

std::basic_ofstream< _CharT, _Traits >, 1378
std::basic_ostream< _CharT, _Traits >, 1408
std::basic_ostream< _CharT, _Traits >::sentry, 2922
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1442
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1713
std::ios_base, 2266
begin
__gnu_cxx::_Temporary_buffer< _ForwardIterator, _Tp >, 1019
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 821
__gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >, 3049
__gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >, 948
__gnu_pbds::sample_trie_access_traits, 2855
__gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >, 3118
Filesystem, 162
Numeric Arrays, 222
std, 588
std::_Temporary_buffer< _ForwardIterator, _Tp >, 1021
std::basic_string< _CharT, _Traits, _Alloc >, 1574
std::deque< _Tp, _Alloc >, 2041
std::forward_list< _Tp, _Alloc >, 2147
std::initializer_list< _E >, 2249
std::list< _Tp, _Alloc >, 2335
std::map< _Key, _Tp, _Compare, _Alloc >, 2387
std::match_results< _Bi_iter, _Alloc >, 2412
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2482
std::multiset< _Key, _Compare, _Alloc >, 2510
std::set< _Key, _Compare, _Alloc >, 2937
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3176
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3206
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3234
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3260, 3262
std::vector< _Tp, _Alloc >, 3298
std::vector< bool, _Alloc >, 3319
Bernoulli Distributions, 234
operator<<, 235
operator>>, 235, 237
bernoulli_distribution
std::bernoulli_distribution, 1718
beta
Mathematical Special Functions, 195
std::gamma_distribution< _RealType >, 2180
TR1 Mathematical Special Functions, 249
betaf
Mathematical Special Functions, 196
betal
Mathematical Special Functions, 196
bin_search_tree_.hpp, 3611
binary
std::basic_fstream< _CharT, _Traits >, 1156
std::basic_ifstream< _CharT, _Traits >, 1198
std::basic_ios< _CharT, _Traits >, 1221
std::basic_iostream< _CharT, _Traits >, 1267
std::basic_istream< _CharT, _Traits >, 1305
std::basic_istream< _CharT, _Traits >::sentry, 2894
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1344
std::basic_ofstream< _CharT, _Traits >, 1379
std::basic_ostream< _CharT, _Traits >, 1409
std::basic_ostream< _CharT, _Traits >::sentry, 2922
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1442
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1713
std::ios_base, 2266
Binary Search, 81
binary_search, 82
equal_range, 83
lower_bound, 84
upper_bound, 85
binary_heap_.hpp, 3615
binary_heap_const_iterator_
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1739
binary_heap_point_const_iterator_
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1743
binary_search
Binary Search, 82
bind
Binder Classes, 313
bind1st
Binder Classes, 313
bind2nd
Binder Classes, 313
Binder Classes, 311
bind, 313
bind1st, 313
bind2nd, 313
binders.h, 3384
binomial_heap_.hpp, 3628
binomial_heap_base_.hpp, 3629
bit, 3363
Bit manipulation, 177
bitmap_allocator.h, 3606

_BALLOC_ALIGN_BYTES, 3607
bitset, 3539
std::bitset< _Nb >, 1763, 1764
bool_set, 3766
std::tr2::bool_set, 1771
bool_set.tcc, 3767
boolalpha
std, 589
std::basic_fstream< _CharT, _Traits >, 1156
std::basic_ifstream< _CharT, _Traits >, 1198
std::basic_ios< _CharT, _Traits >, 1221
std::basic_iostream< _CharT, _Traits >, 1268
std::basic_istream< _CharT, _Traits >, 1305
std::basic_istream< _CharT, _Traits >::sentry, 2894
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1344
std::basic_ofstream< _CharT, _Traits >, 1379
std::basic_ostream< _CharT, _Traits >, 1409
std::basic_ostream< _CharT, _Traits >::sentry, 2922
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1442
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1713
std::ios_base, 2267
Boolean Operations Classes, 314
boost_concept_check.h, 3384
Branch-Based, 139
branch_policy.hpp, 3629
bucket
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 981
bucket_count
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3177
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3207
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3235
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3262

c
std::queue< _Tp, _Sequence >, 2778
c++0x_warning.h, 3385
c++allocator.h, 3789
c++config.h, 3790
c++io.h, 3797
c++locale.h, 3797
c++locale_internal.h, 3797
c_str
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 821
std::basic_string< _CharT, _Traits, _Alloc >, 1575
cache_line_size
__gnu_parallel::_Settings, 1010
call_once
Mutexes, 117
std::once_flag, 2685
capacity
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 821
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1488
std::basic_string< _CharT, _Traits, _Alloc >, 1575
std::vector< _Tp, _Alloc >, 3298
std::vector< bool, _Alloc >, 3319
cassert, 3540
cast.h, 3607
category
std::error_code, 2112
std::error_condition, 2114
std::locale, 2351
cbefore_begin
std::forward_list< _Tp, _Alloc >, 2147
cbegin
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 821
std, 589
std::basic_string< _CharT, _Traits, _Alloc >, 1575
std::deque< _Tp, _Alloc >, 2042
std::forward_list< _Tp, _Alloc >, 2147
std::list< _Tp, _Alloc >, 2335
std::map< _Key, _Tp, _Compare, _Alloc >, 2387
std::match_results< _Bi_iter, _Alloc >, 2412
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2482
std::multiset< _Key, _Compare, _Alloc >, 2510
std::set< _Key, _Compare, _Alloc >, 2937
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3177
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3207
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3235
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3262
std::vector< _Tp, _Alloc >, 3298
std::vector< bool, _Alloc >, 3319
cc_hash_max_collision_check_resize_trigger
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1777
cc_hash_max_collision_check_resize_trigger_imp.hpp, 3647
cc_hash_table
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >, 1781–1784
cc_ht_map_.hpp, 3630

ccomplex, 3540
cctype, 3541
ceil
    chrono.h, 3390
    Time, 366, 367
cend
    __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 821
    std, 589
    std::basic_string< _CharT, _Traits, _Alloc >, 1576
    std::deque< _Tp, _Alloc >, 2042
    std::forward_list< _Tp, _Alloc >, 2147
    std::list< _Tp, _Alloc >, 2335
    std::map< _Key, _Tp, _Compare, _Alloc >, 2387
    std::match_results< _Bi_iter, _Alloc >, 2413
    std::multimap< _Key, _Tp, _Compare, _Alloc >, 2482
    std::multiset< _Key, _Compare, _Alloc >, 2510
    std::set< _Key, _Compare, _Alloc >, 2937
    std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3177
    std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3207
    std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3235
    std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3263
    std::vector< _Tp, _Alloc >, 3298
    std::vector< bool, _Alloc >, 3319
cerr
    std, 646
cerrno, 3541
cfenv, 3541, 3542
cfloat, 3542
char_traits.h, 3385
char_type
    std::__ctype_abstract_base< _CharT >, 751
    std::__cxx11::collate< _CharT >, 1857
    std::__cxx11::collate_byname< _CharT >, 1862
    std::basic_ios< _CharT, _Traits >, 1207
    std::basic_streambuf< _CharT, _Traits >, 1459
    std::ctype< char >, 1926
    std::ctype< wchar_t >, 1960
    std::istreambuf_iterator< _CharT, _Traits >, 2294
    std::messages< _CharT >, 2431
    std::money_get< _CharT, _InIter >, 2442
    std::money_put< _CharT, _OutIter >, 2446
    std::moneypunct< _CharT, _Intl >, 2451
    std::num_get< _CharT, _InIter >, 2547
    std::num_put< _CharT, _OutIter >, 2562
    std::numpunct< _CharT >, 2674
    std::ostream_iterator< _Tp, _CharT, _Traits >, 2688
    std::ostreambuf_iterator< _CharT, _Traits >, 2691
    std::time_get< _CharT, _InIter >, 3064
    std::time_put< _CharT, _OutIter >, 3086
    std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3331
charconv, 3542
charconv.h, 3386
chars_format
    std, 580
checkers.h, 3727
chrono, 3544, 3547
chrono.h, 3386
    ceil, 3390
chrono_io.h, 3390
cin
    std, 646
cinttypes, 3547
ciso646, 3547
clamp
    Sorting, 62, 63
classic
    std::locale, 2353
classic_table
    std::ctype< char >, 1927
    std::ctype_byname< char >, 1993
clear
    __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 822
    std::basic_fstream< _CharT, _Traits >, 1120
    std::basic_ifstream< _CharT, _Traits >, 1171
    std::basic_ios< _CharT, _Traits >, 1210
    std::basic_iostream< _CharT, _Traits >, 1233
    std::basic_istream< _CharT, _Traits >, 1279
    std::basic_istream< _CharT, _Traits >::sentry, 2871
    std::basic_istringstream< _CharT, _Traits, _Alloc >, 1318
    std::basic_ofstream< _CharT, _Traits >, 1356
    std::basic_ostream< _CharT, _Traits >, 1389
    std::basic_ostream< _CharT, _Traits >::sentry, 2905
    std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1421
    std::basic_string< _CharT, _Traits, _Alloc >, 1576
    std::basic_stringstream< _CharT, _Traits, _Alloc >, 1678
    std::deque< _Tp, _Alloc >, 2042
    std::error_condition, 2114
    std::experimental::fundamentals_v1::any, 1065
    std::forward_list< _Tp, _Alloc >, 2147
    std::list< _Tp, _Alloc >, 2335
    std::map< _Key, _Tp, _Compare, _Alloc >, 2388
    std::multimap< _Key, _Tp, _Compare, _Alloc >, 2482
    std::multiset< _Key, _Compare, _Alloc >, 2510
    std::set< _Key, _Compare, _Alloc >, 2937
    std::tr2::dynamic_bitset< _WordT, _Alloc >, 2075
    std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3178

std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3208
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3236
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3263
std::vector< _Tp, _Alloc >, 3298
std::vector< bool, _Alloc >, 3319
climits, 3548
clocale, 3548
clock_cast
Time, 367
clog
std, 646
close
__gnu_cxx::enc_filebuf< _CharT >, 2087
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2992
std::basic_filebuf< _CharT, _Traits >, 1094
std::basic_fstream< _CharT, _Traits >, 1120
std::basic_ifstream< _CharT, _Traits >, 1171
std::basic_ofstream< _CharT, _Traits >, 1356
cmath, 3548, 3559, 3560
cmp_fn
__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >, 3095
cmp_fn_imps.hpp, 3630
code
std::regex_error, 2816
codecvt, 3562
codecvt.h, 3394
codecvt_specializations.h, 3608
collate
std::__cxx11::collate< _CharT >, 1857, 1858
std::locale, 2357
std::regex_constants, 715
combine
std::locale, 2353
common_reference_t
Metaprogramming, 351
common_type_t
Metaprogramming, 351
comp_ellint_1
Mathematical Special Functions, 196
TR1 Mathematical Special Functions, 249
comp_ellint_1f
Mathematical Special Functions, 197
comp_ellint_1l
Mathematical Special Functions, 197
comp_ellint_2
Mathematical Special Functions, 197
TR1 Mathematical Special Functions, 249
comp_ellint_2f
Mathematical Special Functions, 198
comp_ellint_2l
Mathematical Special Functions, 198
comp_ellint_3
Mathematical Special Functions, 198
TR1 Mathematical Special Functions, 249
comp_ellint_3f
Mathematical Special Functions, 199
comp_ellint_3l
Mathematical Special Functions, 199
compare, 3801
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 822–824
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1488–1494
std::__cxx11::collate< _CharT >, 1858
std::basic_string< _CharT, _Traits, _Alloc >, 1576–1581
std::sub_match< _BiIter >, 3036
compare_three_way_result_t
std, 579
Comparison Classes, 314
compatibility.h, 3798
compiletime_settings.h, 3727
_GLIBCXX_CALL, 3727
_GLIBCXX_PARALLEL_ASSERTIONS, 3728
_GLIBCXX_RANDOM_SHUFFLE_CONSIDER_L1, 3728
_GLIBCXX_RANDOM_SHUFFLE_CONSIDER_TLB, 3728
_GLIBCXX_SCALE_DOWN_FPU, 3728
_GLIBCXX_VERBOSE_LEVEL, 3728
complex, 3563, 3567
std::complex< _Tp >, 1867
std::complex< double >, 1869
std::complex< float >, 1871
std::complex< long double >, 1874
Complex Numbers, 177
abs, 181
arg, 181
conj, 181
cos, 181
cosh, 181
exp, 181
fabs, 181
log, 181
log10, 182
norm, 182
operator<<, 185
operator>>, 185
operator+, 183
operator+=, 183
operator-, 183, 184
operator-=, 184
operator/, 184
operator/=, 184, 185

operator=, 185
operator==, 185
operator*, 182
operator*=, 182
polar, 186
pow, 186
sin, 186
sinh, 187
sqrt, 187
tan, 187
tanh, 187
complex.h, 3568
compose1
SGI, 153
compose2
SGI, 153
concept_check.h, 3394
concepts, 3568
concurrence.h, 3608
Concurrency, 111
cond_dealtor.hpp, 3634
cond_key_dtor_entry_dealtor.hpp, 3631
Condition Variables, 111
cv_status, 112
condition_variable, 3568
conditional_t
Metaprogramming, 351
conf_hyperg
Mathematical Special Functions, 199
std::tr1, 724
conf_hypergf
Mathematical Special Functions, 200
std::tr1, 725
conf_hypergl
Mathematical Special Functions, 200
std::tr1, 725
conj
Complex Numbers, 181
Const-propagating wrapper, 282
const_iterator
__gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >, 3118
std::set< _Key, _Compare, _Alloc >, 2932
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3171
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3201
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3230
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3256
const_iterator.hpp, 3615, 3616
const_local_iterator
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3171
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3202
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3230
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3256
const_pointer
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1738
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1742
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >, 2305
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_ite
Node, _Alloc >, 2309
std::allocator_traits< _Alloc >, 1032
std::allocator_traits< allocator< _Tp > >, 1038
std::allocator_traits< allocator< void > >, 1047
std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1055
std::set< _Key, _Compare, _Alloc >, 2932
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3171
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3202
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3230
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3256
const_pointer_cast
Pointer Abstractions, 337
std, 589
const_reference
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1723
__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1728
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1738
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1742
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >, 2305
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_ite
Node, _Alloc >, 2309
std::set< _Key, _Compare, _Alloc >, 2932
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3171
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3202
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3230

std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3257
const_reverse_iterator
std::set< _Key, _Compare, _Alloc >, 2932
const_void_pointer
__gnu_cxx::__alloc_traits< _Alloc, typename >, 734
std::allocator_traits< _Alloc >, 1032
std::allocator_traits< allocator< _Tp > >, 1038
std::allocator_traits< allocator< void > >, 1047
std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1056
constant0
SGI, 153
constant1
SGI, 153
constant2
SGI, 154
construct
__gnu_cxx::__alloc_traits< _Alloc, typename >, 736
std::allocator_traits< _Alloc >, 1034
std::allocator_traits< allocator< _Tp > >, 1042
std::allocator_traits< allocator< void > >, 1051
std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1059, 1060
constructor_destructor_fn_imps.hpp, 3631
constructor_destructor_no_store_hash_fn_imps.hpp, 3631
constructor_destructor_store_hash_fn_imps.hpp, 3631
constructors_destructor_fn_imps.hpp, 3616, 3617
container_base_dispatch.hpp, 3634
container_type
std::back_insert_iterator< _Container >, 1077
std::front_insert_iterator< _Container >, 2162
std::insert_iterator< _Container >, 2253
Containers, 120, 139
contains
std::map< _Key, _Tp, _Compare, _Alloc >, 2388
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2483
std::multiset< _Key, _Compare, _Alloc >, 2510, 2511
std::set< _Key, _Compare, _Alloc >, 2937, 2938
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3178
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3208
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3236
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3263
converted
std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >, 3350
copy
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 825
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1494
Mutating, 21
std::basic_string< _CharT, _Traits, _Alloc >, 1582
copy_backward
Mutating, 21
copy_if
Mutating, 22
copy_n
Mutating, 22
SGI, 154
copy_options
Filesystem, 161
Filesystem TS, 279
copyable_function
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>, 1906, 1907
copyfmt
std::basic_fstream< _CharT, _Traits >, 1120
std::basic_ifstream< _CharT, _Traits >, 1171
std::basic_ios< _CharT, _Traits >, 1210
std::basic_iostream< _CharT, _Traits >, 1233
std::basic_istream< _CharT, _Traits >, 1279
std::basic_istream< _CharT, _Traits >::sentry, 2871
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1318
std::basic_ofstream< _CharT, _Traits >, 1356
std::basic_ostream< _CharT, _Traits >, 1390
std::basic_ostream< _CharT, _Traits >::sentry, 2905
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1423
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1678
cos
Complex Numbers, 181
math.h, 3701
cosh
Complex Numbers, 181
math.h, 3702
count
Non-Mutating, 45
std::bitset< _Nb >, 1765
std::map< _Key, _Tp, _Compare, _Alloc >, 2388, 2389
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2483, 2484
std::multiset< _Key, _Compare, _Alloc >, 2511
std::set< _Key, _Compare, _Alloc >, 2938
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2075
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3179
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3209

std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3237
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3264
count_if
Non-Mutating, 46
count_minimal_n
__gnu_parallel::_Settings, 1011
cout
std, 646
cow_string.h, 3394
cpp_type_traits.h, 3395
cpu_defines.h, 3798
cpyfunc_impl.h, 3395
crbegin
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 825
std, 589
std::basic_string< _CharT, _Traits, _Alloc >, 1583
std::deque< _Tp, _Alloc >, 2042
std::list< _Tp, _Alloc >, 2335
std::map< _Key, _Tp, _Compare, _Alloc >, 2389
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2484
std::multiset< _Key, _Compare, _Alloc >, 2511
std::set< _Key, _Compare, _Alloc >, 2939
std::vector< _Tp, _Alloc >, 3299
std::vector< bool, _Alloc >, 3320
cref
std::reference_wrapper< _Tp >, 2815
cregex_token_iterator
Regular Expressions, 255
crend
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 825
std, 590
std::basic_string< _CharT, _Traits, _Alloc >, 1583
std::deque< _Tp, _Alloc >, 2042
std::list< _Tp, _Alloc >, 2336
std::map< _Key, _Tp, _Compare, _Alloc >, 2389
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2484
std::multiset< _Key, _Compare, _Alloc >, 2512
std::set< _Key, _Compare, _Alloc >, 2939
std::vector< _Tp, _Alloc >, 3299
std::vector< bool, _Alloc >, 3320
csetjmp, 3569
cshift
Numeric Arrays, 222
csignal, 3569
cstdalign, 3570
cstdarg, 3570
cstdbool, 3570, 3571
cstddef, 3571
cstdint, 3571
cstdio, 3572
cstdlib, 3572
cstring, 3573
csub_match
Regular Expressions, 255
ctgmath, 3573
ctime, 3574
ctype
std::ctype< char >, 1926
std::ctype< wchar_t >, 1960, 1961
std::locale, 2357
ctype_base.h, 3798
ctype_inline.h, 3798
cuchar, 3574
cur
std::basic_fstream< _CharT, _Traits >, 1156
std::basic_ifstream< _CharT, _Traits >, 1199
std::basic_ios< _CharT, _Traits >, 1221
std::basic_iostream< _CharT, _Traits >, 1268
std::basic_istream< _CharT, _Traits >, 1305
std::basic_istream< _CharT, _Traits >::sentry, 2895
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1345
std::basic_ofstream< _CharT, _Traits >, 1379
std::basic_ostream< _CharT, _Traits >, 1409
std::basic_ostream< _CharT, _Traits >::sentry, 2922
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1443
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1713
std::ios_base, 2267
curr_symbol
std::moneypunct< _CharT, _Intl >, 2453
std::moneypunct_byname< _CharT, _Intl >, 2460
current_exception
Exceptions, 129
cv_status
Condition Variables, 112
cwchar, 3574, 3575
cwctype, 3575
cxxabi.h, 3802
__cxa_demangle, 3803
cxxabi_forced.h, 3396
cxxabi_init_exception.h, 3396
cxxabi_tweaks.h, 3799
cyl_bessel_i
Mathematical Special Functions, 200
TR1 Mathematical Special Functions, 249
cyl_bessel_if
Mathematical Special Functions, 201
cyl_bessel_il
Mathematical Special Functions, 201
cyl_bessel_j

Mathematical Special Functions, 201
TR1 Mathematical Special Functions, 249
cyl_bessel_jf
Mathematical Special Functions, 201
cyl_bessel_jl
Mathematical Special Functions, 202
cyl_bessel_k
Mathematical Special Functions, 202
TR1 Mathematical Special Functions, 249
cyl_bessel_kf
Mathematical Special Functions, 202
cyl_bessel_kl
Mathematical Special Functions, 203
cyl_neumann
Mathematical Special Functions, 203
TR1 Mathematical Special Functions, 250
cyl_neumannf
Mathematical Special Functions, 203
cyl_neumannl
Mathematical Special Functions, 204

data
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 826
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1495
std, 590, 591
std::basic_string< _CharT, _Traits, _Alloc >, 1583, 1584
std::vector< _Tp, _Alloc >, 3299
std::vector< bool, _Alloc >, 3320
Data parallel extensions, 294
Data Structure Type, 145
date_order
std::time_get< _CharT, _InIter >, 3064
std::time_get_byname< _CharT, _InIter >, 3075
days
Time, 365
deallocate
__gnu_cxx::__alloc_traits< _Alloc, typename >, 737
std::allocator_traits< _Alloc >, 1035
std::allocator_traits< allocator< _Tp > >, 1043
std::allocator_traits< allocator< void > >, 1051, 1052
std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1060
debug.h, 3576
debug_allocator.h, 3609
debug_fn_imps.hpp, 3617–3619
debug_map_base.hpp, 3635
debug_no_store_hash_fn_imps.hpp, 3632
debug_store_hash_fn_imps.hpp, 3632
dec
std, 591
std::basic_fstream< _CharT, _Traits >, 1156
std::basic_ifstream< _CharT, _Traits >, 1199
std::basic_ios< _CharT, _Traits >, 1221
std::basic_iostream< _CharT, _Traits >, 1268
std::basic_istream< _CharT, _Traits >, 1305
std::basic_istream< _CharT, _Traits >::sentry, 2895
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1345
std::basic_ofstream< _CharT, _Traits >, 1379
std::basic_ostream< _CharT, _Traits >, 1409
std::basic_ostream< _CharT, _Traits >::sentry, 2922
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1443
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1713
std::ios_base, 2267
decay_t
Metaprogramming, 351
decimal, 3592
Decimal Floating-Point Arithmetic, 187
decimal128
std::decimal::decimal128, 2015
decimal32
std::decimal::decimal32, 2017
decimal32_to_long_long
std::decimal, 690
decimal64
std::decimal::decimal64, 2018
decimal_point
std::moneypunct< _CharT, _Intl >, 2453
std::moneypunct_byname< _CharT, _Intl >, 2460
std::numpunct< _CharT >, 2675
std::numpunct_byname< _CharT >, 2679
declare_no_pointers
Pointer Safety and Garbage Collection, 344
declare_reachable
Pointer Safety and Garbage Collection, 344
declval
Utilities, 299
default_delete
std::default_delete< _Tp >, 2020
std::default_delete< _Tp[]>, 2021
default_error_condition
std::error_category, 2111
std::error_code, 2112
default_sentinel
std, 646
defaultfloat
std, 591
defer_lock
Mutexes, 118
denorm_absent
std, 580
denorm_indeterminate

std, 580
denorm_min
std::numeric_limits< _Tp >, 2574
std::numeric_limits< bool >, 2580
std::numeric_limits< char >, 2585
std::numeric_limits< char16_t >, 2590
std::numeric_limits< char32_t >, 2596
std::numeric_limits< double >, 2601
std::numeric_limits< float >, 2606
std::numeric_limits< int >, 2611
std::numeric_limits< long >, 2616
std::numeric_limits< long double >, 2622
std::numeric_limits< long long >, 2627
std::numeric_limits< short >, 2632
std::numeric_limits< signed char >, 2638
std::numeric_limits< unsigned char >, 2643
std::numeric_limits< unsigned int >, 2648
std::numeric_limits< unsigned long >, 2653
std::numeric_limits< unsigned long long >, 2659
std::numeric_limits< unsigned short >, 2664
std::numeric_limits< wchar_t >, 2669
denorm_present
std, 580
densities
std::piecewise_constant_distribution< _RealType >, 2744
std::piecewise_linear_distribution< _RealType >, 2748
Deprecated List, 1
deque, 3601, 3602
std::deque< _Tp, _Alloc >, 2033–2035
deque.tcc, 3396
destroy
__gnu_cxx::__alloc_traits< _Alloc, typename >, 737
std::allocator_traits< _Alloc >, 1035
std::allocator_traits< allocator< _Tp > >, 1043, 1044
std::allocator_traits< allocator< void > >, 1052
std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1061
destroying_delete
std, 646
detected_or
Detection idiom, 285
detected_or_t
Detection idiom, 285
detected_t
Detection idiom, 285
Detection idiom, 284
__cpp_lib_experimental_detect, 285
detected_or, 285
detected_or_t, 285
detected_t, 285
is_detected, 285
is_detected_convertible, 285
is_detected_convertible_v, 285
is_detected_exact, 285
is_detected_exact_v, 286
is_detected_v, 286
void_t, 285
Diagnostics, 123
generic_category, 124
make_error_code, 124
make_error_condition, 124
operator<<, 125
operator<=>, 125
operator==, 125, 126
system_category, 126
difference_type
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1723
__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1728
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1738
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1742
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >, 2305
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_ite
Node, _Alloc >, 2309
std::allocator_traits< _Alloc >, 1033
std::allocator_traits< allocator< _Tp > >, 1038, 1039
std::allocator_traits< allocator< void > >, 1047
std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1056
std::istream_iterator< _Tp, _CharT, _Traits, _Dist >, 2292
std::istreambuf_iterator< _CharT, _Traits >, 2294
std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >, 2300
std::ostream_iterator< _Tp, _CharT, _Traits >, 2688
std::ostreambuf_iterator< _CharT, _Traits >, 2691
std::pointer_traits< _Ptr >, 2755
std::pointer_traits< _Tp * >, 2757
std::raw_storage_iterator< _OutputIterator, _Tp >, 2796
std::set< _Key, _Compare, _Alloc >, 2933
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3171
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3202
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3230
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3257
digits

std::__numeric_limits_base, 788
std::numeric_limits< _Tp >, 2575
std::numeric_limits< bool >, 2581
std::numeric_limits< char >, 2586
std::numeric_limits< char16_t >, 2591
std::numeric_limits< char32_t >, 2597
std::numeric_limits< double >, 2602
std::numeric_limits< float >, 2607
std::numeric_limits< int >, 2612
std::numeric_limits< long >, 2617
std::numeric_limits< long double >, 2623
std::numeric_limits< long long >, 2628
std::numeric_limits< short >, 2633
std::numeric_limits< signed char >, 2639
std::numeric_limits< unsigned char >, 2644
std::numeric_limits< unsigned int >, 2649
std::numeric_limits< unsigned long >, 2654
std::numeric_limits< unsigned long long >, 2660
std::numeric_limits< unsigned short >, 2665
std::numeric_limits< wchar_t >, 2670
digits10
std::__numeric_limits_base, 788
std::numeric_limits< _Tp >, 2576
std::numeric_limits< bool >, 2581
std::numeric_limits< char >, 2586
std::numeric_limits< char16_t >, 2591
std::numeric_limits< char32_t >, 2597
std::numeric_limits< double >, 2602
std::numeric_limits< float >, 2607
std::numeric_limits< int >, 2612
std::numeric_limits< long >, 2617
std::numeric_limits< long double >, 2623
std::numeric_limits< long long >, 2628
std::numeric_limits< short >, 2633
std::numeric_limits< signed char >, 2639
std::numeric_limits< unsigned char >, 2644
std::numeric_limits< unsigned int >, 2649
std::numeric_limits< unsigned long >, 2654
std::numeric_limits< unsigned long long >, 2660
std::numeric_limits< unsigned short >, 2665
std::numeric_limits< wchar_t >, 2670
direct_mask_range_hashing_imp.hpp, 3637
direct_mod_range_hashing_imp.hpp, 3637
directory_options
Filesystem, 161
Filesystem TS, 279
discard
std::discard_block_engine< _RandomNumberEngine, __p, __r >, 2058
std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >, 2243
std::linear_congruential_engine< _UIntType, __a, __c, __m >, 2319
std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >, 2427
std::shuffle_order_engine< _RandomNumberEngine, __k >, 2971
std::subtract_with_carry_engine< _UIntType, __w, __s, __r >, 3041
discard_block_engine
std::discard_block_engine< _RandomNumberEngine, __p, __r >, 2057, 2058
distance
SGI, 154
std, 591
do_allocate
std::pmr::monotonic_buffer_resource, 2467
std::pmr::unsynchronized_pool_resource, 3279
do_always_noconv
std::codecvt< _InternT, _ExternT, _StateT >, 1801
std::codecvt< _InternT, _ExternT, encoding_state >, 1807
std::codecvt< char, char, mbstate_t >, 1816
std::codecvt< char16_t, char, mbstate_t >, 1825
std::codecvt< char32_t, char, mbstate_t >, 1834
std::codecvt< wchar_t, char, mbstate_t >, 1843
std::codecvt_byname< _InternT, _ExternT, _StateT >, 1852
do_compare
std::__cxx11::collate< _CharT >, 1858
do_curr_symbol
std::moneypunct< _CharT, _Intl >, 2453
std::moneypunct_byname< _CharT, _Intl >, 2461
do_date_order
std::time_get< _CharT, _InIter >, 3064
std::time_get_byname< _CharT, _InIter >, 3075
do_deallocate
std::pmr::monotonic_buffer_resource, 2467
std::pmr::unsynchronized_pool_resource, 3279
do_decimal_point
std::moneypunct< _CharT, _Intl >, 2453
std::moneypunct_byname< _CharT, _Intl >, 2461
std::numpunct< _CharT >, 2675
std::numpunct_byname< _CharT >, 2680
do_encoding
std::codecvt< _InternT, _ExternT, _StateT >, 1801
std::codecvt< _InternT, _ExternT, encoding_state >, 1807
std::codecvt< char, char, mbstate_t >, 1816
std::codecvt< char16_t, char, mbstate_t >, 1825
std::codecvt< char32_t, char, mbstate_t >, 1834
std::codecvt< wchar_t, char, mbstate_t >, 1843
std::codecvt_byname< _InternT, _ExternT, _StateT >, 1852
do_falsename
std::numpunct< _CharT >, 2676
std::numpunct_byname< _CharT >, 2680

do_frac_digits
std::moneypunct< _CharT, _Intl >, 2454
std::moneypunct_byname< _CharT, _Intl >, 2461
do_get
std::messages< _CharT >, 2432
std::messages_byname< _CharT >, 2434
std::money_get< _CharT, _InIter >, 2443
std::num_get< _CharT, _InIter >, 2547–2553
std::time_get< _CharT, _InIter >, 3065
std::time_get_byname< _CharT, _InIter >, 3075
do_get_date
std::time_get< _CharT, _InIter >, 3065
std::time_get_byname< _CharT, _InIter >, 3076
do_get_monthname
std::time_get< _CharT, _InIter >, 3066
std::time_get_byname< _CharT, _InIter >, 3076
do_get_time
std::time_get< _CharT, _InIter >, 3067
std::time_get_byname< _CharT, _InIter >, 3077
do_get_weekday
std::time_get< _CharT, _InIter >, 3067
std::time_get_byname< _CharT, _InIter >, 3078
do_get_year
std::time_get< _CharT, _InIter >, 3068
std::time_get_byname< _CharT, _InIter >, 3078
do_grouping
std::moneypunct< _CharT, _Intl >, 2454
std::moneypunct_byname< _CharT, _Intl >, 2462
std::numpunct< _CharT >, 2676
std::numpunct_byname< _CharT >, 2680
do_hash
std::__cxx11::collate< _CharT >, 1859
do_in
std::codecvt< _InternT, _ExternT, _StateT >, 1802
std::codecvt< _InternT, _ExternT, encoding_state >, 1807, 1808
std::codecvt< char, char, mbstate_t >, 1816
std::codecvt< char16_t, char, mbstate_t >, 1825
std::codecvt< char32_t, char, mbstate_t >, 1834
std::codecvt< wchar_t, char, mbstate_t >, 1843
std::codecvt_byname< _InternT, _ExternT, _StateT >, 1852
do_is
std::__ctype_abstract_base< _CharT >, 751
std::ctype< _CharT >, 1912
std::ctype< char >, 1927, 1928
std::ctype< wchar_t >, 1961, 1962
std::ctype_byname< _CharT >, 1980
std::ctype_byname< char >, 1993, 1994
do_is_equal
std::pmr::monotonic_buffer_resource, 2467
std::pmr::unsynchronized_pool_resource, 3279
do_length
std::codecvt< _InternT, _ExternT, _StateT >, 1802
std::codecvt< _InternT, _ExternT, encoding_state >, 1808
std::codecvt< char, char, mbstate_t >, 1817
std::codecvt< char16_t, char, mbstate_t >, 1825, 1826
std::codecvt< char32_t, char, mbstate_t >, 1834, 1835
std::codecvt< wchar_t, char, mbstate_t >, 1844
std::codecvt_byname< _InternT, _ExternT, _StateT >, 1853
do_max_length
std::codecvt< _InternT, _ExternT, _StateT >, 1802
std::codecvt< _InternT, _ExternT, encoding_state >, 1808
std::codecvt< char, char, mbstate_t >, 1817
std::codecvt< char16_t, char, mbstate_t >, 1826
std::codecvt< char32_t, char, mbstate_t >, 1835
std::codecvt< wchar_t, char, mbstate_t >, 1844
std::codecvt_byname< _InternT, _ExternT, _StateT >, 1853
do_narrow
std::__ctype_abstract_base< _CharT >, 752
std::ctype< _CharT >, 1913
std::ctype< char >, 1929, 1930, 1932
std::ctype< wchar_t >, 1963, 1964
std::ctype_byname< _CharT >, 1980, 1981
std::ctype_byname< char >, 1995, 1996
do_neg_format
std::moneypunct< _CharT, _Intl >, 2454
std::moneypunct_byname< _CharT, _Intl >, 2462
do_negative_sign
std::moneypunct< _CharT, _Intl >, 2454
std::moneypunct_byname< _CharT, _Intl >, 2462
do_out
std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >, 743
std::codecvt< _InternT, _ExternT, _StateT >, 1802
std::codecvt< _InternT, _ExternT, encoding_state >, 1808, 1809
std::codecvt< char, char, mbstate_t >, 1817
std::codecvt< char16_t, char, mbstate_t >, 1826
std::codecvt< char32_t, char, mbstate_t >, 1835
std::codecvt< wchar_t, char, mbstate_t >, 1844
std::codecvt_byname< _InternT, _ExternT, _StateT >, 1853
do_pos_format
std::moneypunct< _CharT, _Intl >, 2455
std::moneypunct_byname< _CharT, _Intl >, 2462
do_positive_sign
std::moneypunct< _CharT, _Intl >, 2455
std::moneypunct_byname< _CharT, _Intl >, 2463
do_put
std::money_put< _CharT, _OutIter >, 2447
std::num_put< _CharT, _OutIter >, 2563–2567

std::time_put< _CharT, _OutIter >, 3086
std::time_put_byname< _CharT, _OutIter >, 3090
do_scan_is
std::__ctype_abstract_base< _CharT >, 753
std::ctype< _CharT >, 1914
std::ctype< char >, 1933
std::ctype< wchar_t >, 1964, 1965
std::ctype_byname< _CharT >, 1981
std::ctype_byname< char >, 1997
do_scan_not
std::__ctype_abstract_base< _CharT >, 753
std::ctype< _CharT >, 1914
std::ctype< char >, 1934
std::ctype< wchar_t >, 1965, 1966
std::ctype_byname< _CharT >, 1982
std::ctype_byname< char >, 1998
do_thousands_sep
std::moneypunct< _CharT, _Intl >, 2455
std::moneypunct_byname< _CharT, _Intl >, 2463
std::numpunct< _CharT >, 2676
std::numpunct_byname< _CharT >, 2680
do_tolower
std::__ctype_abstract_base< _CharT >, 754
std::ctype< _CharT >, 1915
std::ctype< char >, 1935–1937
std::ctype< wchar_t >, 1966–1968
std::ctype_byname< _CharT >, 1982, 1983
std::ctype_byname< char >, 1999, 2000
do_toupper
std::__ctype_abstract_base< _CharT >, 755
std::ctype< _CharT >, 1916
std::ctype< char >, 1937–1939
std::ctype< wchar_t >, 1968, 1969
std::ctype_byname< _CharT >, 1983, 1984
std::ctype_byname< char >, 2000, 2001
do_transform
std::__cxx11::collate< _CharT >, 1859
do_truename
std::numpunct< _CharT >, 2676
std::numpunct_byname< _CharT >, 2681
do_unshift
std::codecvt< _InternT, _ExternT, _StateT >, 1802
std::codecvt< _InternT, _ExternT, encoding_state >, 1809
std::codecvt< char, char, mbstate_t >, 1818
std::codecvt< char16_t, char, mbstate_t >, 1827
std::codecvt< char32_t, char, mbstate_t >, 1836
std::codecvt< wchar_t, char, mbstate_t >, 1845
std::codecvt_byname< _InternT, _ExternT, _StateT >, 1853
do_widen
std::__ctype_abstract_base< _CharT >, 756
std::ctype< _CharT >, 1917
std::ctype< char >, 1940, 1941
std::ctype< wchar_t >, 1970, 1971
std::ctype_byname< _CharT >, 1984
std::ctype_byname< char >, 2002, 2003
duration_cast
Time, 367
Dynamic Bitset., 134
operator!=, 135
operator<<, 136
operator<=, 136
operator>, 136
operator>>, 137
operator>=, 136
operator-, 135
operator&, 135
operator^, 137
operator|, 137
dynamic_bitset, 3767
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2073, 2074
dynamic_bitset.tcc, 3768
dynamic_pointer_cast
Pointer Abstractions, 338
std, 592

e_pos
__gnu_pbds::sample_trie_access_traits, 2855
__gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >, 3119
e_type
__gnu_pbds::sample_trie_access_traits, 2855
__gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >, 3118
eback
__gnu_cxx::enc_filebuf< _CharT >, 2087
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2993
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3012
std::basic_filebuf< _CharT, _Traits >, 1094
std::basic_streambuf< _CharT, _Traits >, 1460
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1654
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3332
ECMAScript
std::regex_constants, 715
egptr
__gnu_cxx::enc_filebuf< _CharT >, 2087
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2993
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3012
std::basic_filebuf< _CharT, _Traits >, 1094
std::basic_streambuf< _CharT, _Traits >, 1460
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1654
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3332

egrep
std::regex_constants, 715
element_type
std::auto_ptr< _Tp >, 1072
std::pointer_traits< _Ptr >, 2755
std::pointer_traits< _Tp * >, 2757
std::shared_ptr< _Tp >, 2961
ellint_1
Mathematical Special Functions, 204
TR1 Mathematical Special Functions, 250
ellint_1f
Mathematical Special Functions, 204
ellint_1l
Mathematical Special Functions, 205
ellint_2
Mathematical Special Functions, 205
TR1 Mathematical Special Functions, 250
ellint_2f
Mathematical Special Functions, 206
ellint_2l
Mathematical Special Functions, 206
ellint_3
Mathematical Special Functions, 206
TR1 Mathematical Special Functions, 250
ellint_3f
Mathematical Special Functions, 207
ellint_3l
Mathematical Special Functions, 207
emplace
std::deque< _Tp, _Alloc >, 2042
std::list< _Tp, _Alloc >, 2336
std::map< _Key, _Tp, _Compare, _Alloc >, 2389
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2484
std::multiset< _Key, _Compare, _Alloc >, 2512
std::set< _Key, _Compare, _Alloc >, 2939
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3179
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3209
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3237
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3264
std::vector< _Tp, _Alloc >, 3299
std::vector< bool, _Alloc >, 3320
emplace_after
std::forward_list< _Tp, _Alloc >, 2147
emplace_front
std::forward_list< _Tp, _Alloc >, 2148
emplace_hint
std::map< _Key, _Tp, _Compare, _Alloc >, 2390
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2485
std::multiset< _Key, _Compare, _Alloc >, 2512
std::set< _Key, _Compare, _Alloc >, 2939
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3181
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3210
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3238
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3265
empty
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 826
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1495
__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >, 1788
__gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >, 2194
std, 592
std::basic_string< _CharT, _Traits, _Alloc >, 1584
std::deque< _Tp, _Alloc >, 2043
std::experimental::fundamentals_v1::any, 1065
std::forward_list< _Tp, _Alloc >, 2148
std::list< _Tp, _Alloc >, 2336
std::map< _Key, _Tp, _Compare, _Alloc >, 2390
std::match_results< _Bi_iter, _Alloc >, 2413
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2485
std::multiset< _Key, _Compare, _Alloc >, 2513
std::priority_queue< _Tp, _Sequence, _Compare >, 2768
std::queue< _Tp, _Sequence >, 2777
std::set< _Key, _Compare, _Alloc >, 2940
std::stack< _Tp, _Sequence >, 2987
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2075
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3181
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3210
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3238
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3265
std::vector< _Tp, _Alloc >, 3299
std::vector< bool, _Alloc >, 3320
enable_if_t
Metaprogramming, 351
enable_special_members.h, 3398
enc_filebuf.h, 3609
end

__gnu_cxx::_Temporary_buffer< _ForwardIterator, _Tp >, 1019
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 826
__gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >, 3049
__gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >, 948
__gnu_pbds::detail::trie_policy_base< Node_CItr, Node_Itr, _ATraits, _Alloc >, 3114
__gnu_pbds::sample_trie_access_traits, 2856
__gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >, 3119
Filesystem, 162
Numeric Arrays, 223
std, 592, 593
std::_Temporary_buffer< _ForwardIterator, _Tp >, 1021
std::basic_fstream< _CharT, _Traits >, 1156
std::basic_ifstream< _CharT, _Traits >, 1199
std::basic_ios< _CharT, _Traits >, 1222
std::basic_iostream< _CharT, _Traits >, 1268
std::basic_istream< _CharT, _Traits >, 1306
std::basic_istream< _CharT, _Traits >::sentry, 2895
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1345
std::basic_ofstream< _CharT, _Traits >, 1379
std::basic_ostream< _CharT, _Traits >, 1409
std::basic_ostream< _CharT, _Traits >::sentry, 2922
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1443
std::basic_string< _CharT, _Traits, _Alloc >, 1584, 1585
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1713
std::deque< _Tp, _Alloc >, 2043
std::forward_list< _Tp, _Alloc >, 2148
std::initializer_list< _E >, 2249
std::ios_base, 2267
std::list< _Tp, _Alloc >, 2336
std::map< _Key, _Tp, _Compare, _Alloc >, 2391
std::match_results< _Bi_iter, _Alloc >, 2413
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2485
std::multiset< _Key, _Compare, _Alloc >, 2513
std::set< _Key, _Compare, _Alloc >, 2940
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3181, 3182
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3210, 3211
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3238, 3239
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3265, 3266
std::vector< _Tp, _Alloc >, 3299, 3300
std::vector< bool, _Alloc >, 3320
endl
std, 593
ends
std, 593
entry_cmp.hpp, 3619
entry_list_fn_imps.hpp, 3632
entry_metadata_base.hpp, 3641
entry_pred.hpp, 3619
eof
std::basic_fstream< _CharT, _Traits >, 1122
std::basic_ifstream< _CharT, _Traits >, 1173
std::basic_ios< _CharT, _Traits >, 1210
std::basic_iostream< _CharT, _Traits >, 1233
std::basic_istream< _CharT, _Traits >, 1279
std::basic_istream< _CharT, _Traits >::sentry, 2872
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1319
std::basic_ofstream< _CharT, _Traits >, 1358
std::basic_ostream< _CharT, _Traits >, 1390
std::basic_ostream< _CharT, _Traits >::sentry, 2905
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1423
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1679
eofbit
std::basic_fstream< _CharT, _Traits >, 1157
std::basic_ifstream< _CharT, _Traits >, 1199
std::basic_ios< _CharT, _Traits >, 1222
std::basic_iostream< _CharT, _Traits >, 1268
std::basic_istream< _CharT, _Traits >, 1306
std::basic_istream< _CharT, _Traits >::sentry, 2895
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1345
std::basic_ofstream< _CharT, _Traits >, 1379
std::basic_ostream< _CharT, _Traits >, 1409
std::basic_ostream< _CharT, _Traits >::sentry, 2923
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1443
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1714
std::ios_base, 2267
epptr
__gnu_cxx::enc_filebuf< _CharT >, 2087
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2993
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3013
std::basic_filebuf< _CharT, _Traits >, 1094
std::basic_streambuf< _CharT, _Traits >, 1461
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1654
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3332
epsilon

std::numeric_limits< _Tp >, 2574
std::numeric_limits< bool >, 2580
std::numeric_limits< char >, 2585
std::numeric_limits< char16_t >, 2590
std::numeric_limits< char32_t >, 2596
std::numeric_limits< double >, 2601
std::numeric_limits< float >, 2606
std::numeric_limits< int >, 2611
std::numeric_limits< long >, 2616
std::numeric_limits< long double >, 2622
std::numeric_limits< long long >, 2627
std::numeric_limits< short >, 2632
std::numeric_limits< signed char >, 2638
std::numeric_limits< unsigned char >, 2643
std::numeric_limits< unsigned int >, 2648
std::numeric_limits< unsigned long >, 2653
std::numeric_limits< unsigned long long >, 2659
std::numeric_limits< unsigned short >, 2664
std::numeric_limits< wchar_t >, 2669
eq_by_less.hpp, 3635
equal
Non-Mutating, 46, 47
std::istreambuf_iterator< _CharT, _Traits >, 2296
equal_range
Binary Search, 83
std::map< _Key, _Tp, _Compare, _Alloc >, 2391, 2392
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2486, 2487
std::multiset< _Key, _Compare, _Alloc >, 2513, 2514
std::set< _Key, _Compare, _Alloc >, 2940, 2941
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3182, 3183
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3211, 3212
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3239, 3240
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3266, 3268
equally_split.h, 3728
equals
std::tr2::bool_set, 1771
equivalent
std::error_category, 2111
erase
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 826, 827
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1495, 1496
std::basic_string< _CharT, _Traits, _Alloc >, 1585, 1587, 1588
std::deque< _Tp, _Alloc >, 2043
std::list< _Tp, _Alloc >, 2336, 2337
std::map< _Key, _Tp, _Compare, _Alloc >, 2393
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2487, 2488
std::multiset< _Key, _Compare, _Alloc >, 2515
std::set< _Key, _Compare, _Alloc >, 2942, 2943
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3184, 3185
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3213, 3214
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3240–3242
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3270, 3271
std::vector< _Tp, _Alloc >, 3300
std::vector< bool, _Alloc >, 3321
erase_after
std::chrono::tzdb_list, 3137
std::forward_list< _Tp, _Alloc >, 2149
erase_can_throw
__gnu_pbds::container_traits< Cntnr >, 1899
erase_fn_imps.hpp, 3619–3621
erase_if.h, 3398
erase_no_store_hash_fn_imps.hpp, 3632
erase_store_hash_fn_imps.hpp, 3632
error_backref
std::regex_constants, 711
error_badbrace
std::regex_constants, 711
error_badrepeat
std::regex_constants, 711
error_brace
std::regex_constants, 711
error_brack
std::regex_constants, 711
error_code
std::error_code, 2112
error_collate
std::regex_constants, 711
error_complexity
std::regex_constants, 711
error_condition
std::error_condition, 2114
error_constants.h, 3799
error_ctype
std::regex_constants, 711
error_escape
std::regex_constants, 712
error_paren
std::regex_constants, 712
error_range
std::regex_constants, 712
error_space
std::regex_constants, 712
error_stack
std::regex_constants, 712

error_type
std::regex_constants, 710
event
std::basic_fstream< _CharT, _Traits >, 1118
std::basic_ifstream< _CharT, _Traits >, 1169
std::basic_ios< _CharT, _Traits >, 1209
std::basic_iostream< _CharT, _Traits >, 1232
std::basic_istream< _CharT, _Traits >, 1278
std::basic_istream< _CharT, _Traits >::sentry, 2870
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1317
std::basic_ofstream< _CharT, _Traits >, 1354
std::basic_ostream< _CharT, _Traits >, 1389
std::basic_ostream< _CharT, _Traits >::sentry, 2903
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1420
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1677
std::ios_base, 2260
event_callback
std::basic_fstream< _CharT, _Traits >, 1117
std::basic_ifstream< _CharT, _Traits >, 1168
std::basic_ios< _CharT, _Traits >, 1207
std::basic_iostream< _CharT, _Traits >, 1231
std::basic_istream< _CharT, _Traits >, 1277
std::basic_istream< _CharT, _Traits >::sentry, 2869
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1314
std::basic_ofstream< _CharT, _Traits >, 1353
std::basic_ostream< _CharT, _Traits >, 1387
std::basic_ostream< _CharT, _Traits >::sentry, 2902
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1419
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1675
std::ios_base, 2259
exception, 3804
exception.h, 3398
exception.hpp, 3654
exception_defines.h, 3399
exception_ptr.h, 3399
rethrow_exception, 3399
Exceptions, 127, 143
__verbose_terminate_handler, 129
current_exception, 129
fmtflags, 132
get_terminate, 129
get_unexpected, 129
make_exception_ptr, 130
rethrow_exception, 130
rethrow_if_nested, 130
set_terminate, 130
set_unexpected, 131
terminate, 131
terminate_handler, 128
throw_with_nested, 131
uncaught_exception, 131
unexpected, 132
unexpected_handler, 128
exceptions
std::basic_fstream< _CharT, _Traits >, 1122
std::basic_ifstream< _CharT, _Traits >, 1173
std::basic_ios< _CharT, _Traits >, 1211
std::basic_iostream< _CharT, _Traits >, 1233, 1234
std::basic_istream< _CharT, _Traits >, 1280
std::basic_istream< _CharT, _Traits >::sentry, 2872
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1319
std::basic_ofstream< _CharT, _Traits >, 1358
std::basic_ostream< _CharT, _Traits >, 1390, 1391
std::basic_ostream< _CharT, _Traits >::sentry, 2905
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1423
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1679
exclusive_scan
Generalized Numeric operations, 8, 9
exp
Complex Numbers, 181
math.h, 3702
expected, 3603
expint
Mathematical Special Functions, 207
TR1 Mathematical Special Functions, 250
expintf
Mathematical Special Functions, 208
expintl
Mathematical Special Functions, 208
exponential_distribution
std::exponential_distribution< _RealType >, 2120
extc++.h, 3800
extended
std::regex_constants, 716
Extensions, 133
external_load_access
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1777
__gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >, 2225
__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >, 2235
extptr_allocator.h, 3609

fabs
Complex Numbers, 181
math.h, 3702
std, 593

facet
std::locale::facet, 2127
fail
std::basic_fstream< _CharT, _Traits >, 1123
std::basic_ifstream< _CharT, _Traits >, 1174
std::basic_ios< _CharT, _Traits >, 1211
std::basic_iostream< _CharT, _Traits >, 1234
std::basic_istream< _CharT, _Traits >, 1280
std::basic_istream< _CharT, _Traits >::sentry, 2873
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1320
std::basic_ofstream< _CharT, _Traits >, 1359
std::basic_ostream< _CharT, _Traits >, 1391
std::basic_ostream< _CharT, _Traits >::sentry, 2906
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1424
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1680
failbit
std::basic_fstream< _CharT, _Traits >, 1157
std::basic_ifstream< _CharT, _Traits >, 1199
std::basic_ios< _CharT, _Traits >, 1222
std::basic_iostream< _CharT, _Traits >, 1268
std::basic_istream< _CharT, _Traits >, 1306
std::basic_istream< _CharT, _Traits >::sentry, 2895
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1345
std::basic_ofstream< _CharT, _Traits >, 1379
std::basic_ostream< _CharT, _Traits >, 1409
std::basic_ostream< _CharT, _Traits >::sentry, 2923
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1443
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1714
std::ios_base, 2267
failed
std::ostreambuf_iterator< _CharT, _Traits >, 2692
false_type
Metaprogramming, 351
falsename
std::numpunct< _CharT >, 2677
std::numpunct_byname< _CharT >, 2681
fd
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2993
features.h, 3729
_GLIBCXX_BAL_QUICKSORT, 3729
_GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS, 3729
_GLIBCXX_FIND_EQUAL_SPLIT, 3729
_GLIBCXX_FIND_GROWING_BLOCKS, 3729
_GLIBCXX_MERGESORT, 3730
_GLIBCXX_QUICKSORT, 3730
_GLIBCXX_TREE_DYNAMIC_BALANCING, 3730
_GLIBCXX_TREE_FULL_COPY, 3730
_GLIBCXX_TREE_INITIAL_SPLITTING, 3730
fenv.h, 3680
file
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2994
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3013
file_time_type
Filesystem, 161
Filesystem TS, 279
file_type
Filesystem, 162
Filesystem TS, 279
filebuf
I/O, 165
Filesystem, 157
begin, 162
copy_options, 161
directory_options, 161
end, 162
file_time_type, 161
file_type, 162
perm_options, 162
perms, 162
u8path, 162, 163
filesystem, 3680
Filesystem TS, 274
copy_options, 279
directory_options, 279
file_time_type, 279
file_type, 279
perms, 279
fill
Mutating, 23
std::basic_fstream< _CharT, _Traits >, 1123
std::basic_ifstream< _CharT, _Traits >, 1174
std::basic_ios< _CharT, _Traits >, 1212
std::basic_iostream< _CharT, _Traits >, 1234, 1235
std::basic_istream< _CharT, _Traits >, 1281
std::basic_istream< _CharT, _Traits >::sentry, 2873
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1320
std::basic_ofstream< _CharT, _Traits >, 1359
std::basic_ostream< _CharT, _Traits >, 1391, 1392
std::basic_ostream< _CharT, _Traits >::sentry, 2906
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1424
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1680
fill_minimal_n
__gnu_parallel::_Settings, 1011
fill_n
Mutating, 23
find

__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 827–829
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1497–1501
Non-Mutating, 48
std::basic_string< _CharT, _Traits, _Alloc >, 1588–1591
std::map< _Key, _Tp, _Compare, _Alloc >, 2394, 2395
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2489, 2490
std::multiset< _Key, _Compare, _Alloc >, 2516, 2517
std::set< _Key, _Compare, _Alloc >, 2943, 2944
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3185, 3186
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3214, 3215
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3242, 3244
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3271, 3272
find.h, 3731
find_by_order
__gnu_pbds::tree_order_statistics_node_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc >, 3098
__gnu_pbds::trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >, 3112
find_end
Non-Mutating, 48, 49
find_first
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2075
find_first_not_of
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 829, 830
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1501–1505
std::basic_string< _CharT, _Traits, _Alloc >, 1592–1594
find_first_of
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 831, 832
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1506–1510
Non-Mutating, 50
std::basic_string< _CharT, _Traits, _Alloc >, 1595–1597
find_fn_imps.hpp, 3621, 3622
find_if
Non-Mutating, 51
find_if_not
Non-Mutating, 51
find_increasing_factor
__gnu_parallel::_Settings, 1011
find_initial_block_size
__gnu_parallel::_Settings, 1011
find_last_not_of
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 832–834
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1510–1514
std::basic_string< _CharT, _Traits, _Alloc >, 1598–1600
find_last_of
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 834, 835
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1514–1518
std::basic_string< _CharT, _Traits, _Alloc >, 1601–1603
find_maximum_block_size
__gnu_parallel::_Settings, 1011
find_next
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2075
find_no_store_hash_fn_imps.hpp, 3636
find_scale_factor
__gnu_parallel::_Settings, 1011
find_selectors.h, 3731
find_sequential_search_size
__gnu_parallel::_Settings, 1011
find_store_hash_fn_imps.hpp, 3632, 3633
first
__gnu_cxx::pair< _T1, _T2 >, 2711
__gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >, 899
std::pair< _T1, _T2 >, 2715
first_argument_type
__gnu_cxx::equal_to< _Tp >, 2108
__gnu_cxx::project1st< _Arg1, _Arg2 >, 2770
__gnu_cxx::project2nd< _Arg1, _Arg2 >, 2771
__gnu_parallel::_EqualFromLess< _T1, _T2, _Compare >, 876
__gnu_parallel::_EqualTo< _T1, _T2 >, 877
__gnu_parallel::_Less< _T1, _T2 >, 903
__gnu_parallel::_Lexicographic< _T1, _T2, _Compare >, 904
__gnu_parallel::_LexicographicReverse< _T1, _T2, _Compare >, 905
__gnu_parallel::_Multiplies< _Tp1, _Tp2, _Result >, 931
__gnu_parallel::_Plus< _Tp1, _Tp2, _Result >, 943
std::binary_function< _Arg1, _Arg2, _Result >, 1735
std::binary_negate< _Predicate >, 1745
std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >, 1882
std::const_mem_fun1_t< _Ret, _Tp, _Arg >, 1883
std::divides< _Tp >, 2065
std::divides< void >, 2066
std::equal_to< _Tp >, 2109

std::experimental::fundamentals_v2::owner_less< shared_ptr< _Tp > >, 2704
std::experimental::fundamentals_v2::owner_less< weak_ptr< _Tp > >, 2706
std::greater< _Tp >, 2197
std::greater< void >, 2199
std::greater_equal< _Tp >, 2200
std::greater_equal< void >, 2201
std::less< _Tp >, 2313
std::less_equal< _Tp >, 2315
std::less_equal< void >, 2316
std::logical_and< _Tp >, 2360
std::logical_and< void >, 2361
std::logical_or< _Tp >, 2364
std::logical_or< void >, 2365
std::mem_fun1_ref_t< _Ret, _Tp, _Arg >, 2420
std::mem_fun1_t< _Ret, _Tp, _Arg >, 2421
std::minus< _Tp >, 2435
std::minus< void >, 2436
std::modulus< _Tp >, 2438
std::modulus< void >, 2439
std::multiplies< _Tp >, 2500
std::multiplies< void >, 2501
std::not_equal_to< _Tp >, 2540
std::not_equal_to< void >, 2542
std::owner_less< shared_ptr< _Tp > >, 2705
std::owner_less< void >, 2706
std::owner_less< weak_ptr< _Tp > >, 2707
std::plus< _Tp >, 2751
std::pointer_to_binary_function< _Arg1, _Arg2, _Result >, 2753
first_type
__gnu_cxx::pair< _T1, _T2 >, 2710
__gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >, 895, 896
std::pair< _T1, _T2 >, 2713
fixed
std, 593
std::basic_fstream< _CharT, _Traits >, 1157
std::basic_ifstream< _CharT, _Traits >, 1199
std::basic_ios< _CharT, _Traits >, 1222
std::basic_iostream< _CharT, _Traits >, 1269
std::basic_istream< _CharT, _Traits >, 1306
std::basic_istream< _CharT, _Traits >::sentry, 2895
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1345
std::basic_ofstream< _CharT, _Traits >, 1380
std::basic_ostream< _CharT, _Traits >, 1410
std::basic_ostream< _CharT, _Traits >::sentry, 2923
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1443
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1714
std::ios_base, 2268
flags
std::basic_fstream< _CharT, _Traits >, 1123, 1124
std::basic_ifstream< _CharT, _Traits >, 1174, 1175
std::basic_ios< _CharT, _Traits >, 1212
std::basic_iostream< _CharT, _Traits >, 1235
std::basic_istream< _CharT, _Traits >, 1281
std::basic_istream< _CharT, _Traits >::sentry, 2873
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1320, 1321
std::basic_ofstream< _CharT, _Traits >, 1359, 1360
std::basic_ostream< _CharT, _Traits >, 1392
std::basic_ostream< _CharT, _Traits >::sentry, 2907
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1425
std::basic_regex< _Ch_type, _Rx_traits >, 1453
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1680, 1681
std::ios_base, 2261
flat_map, 3681
flat_set, 3681
flip
std::bitset< _Nb >, 1765
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2076
float_denorm_style
std, 580
float_round_style
std, 580
floatfield
std::basic_fstream< _CharT, _Traits >, 1157
std::basic_ifstream< _CharT, _Traits >, 1200
std::basic_ios< _CharT, _Traits >, 1222
std::basic_iostream< _CharT, _Traits >, 1269
std::basic_istream< _CharT, _Traits >, 1306
std::basic_istream< _CharT, _Traits >::sentry, 2895
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1346
std::basic_ofstream< _CharT, _Traits >, 1380
std::basic_ostream< _CharT, _Traits >, 1410
std::basic_ostream< _CharT, _Traits >::sentry, 2923
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1444
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1714
std::ios_base, 2268
floor
Time, 368
flush
std, 594
std::basic_fstream< _CharT, _Traits >, 1124
std::basic_iostream< _CharT, _Traits >, 1235
std::basic_ofstream< _CharT, _Traits >, 1360
std::basic_ostream< _CharT, _Traits >, 1392
std::basic_ostream< _CharT, _Traits >::sentry, 2907

std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1425
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1681
fmtflags
Exceptions, 132
std::basic_istream< _CharT, _Traits >::sentry, 2895
std::basic_ostream< _CharT, _Traits >::sentry, 2923
for_each
Non-Mutating, 52
for_each.h, 3731
for_each_minimal_n
__gnu_parallel::_Settings, 1011
for_each_n
Non-Mutating, 52
for_each_selectors.h, 3732
format, 3681
std::filesystem::path, 2741
std::match_results< _Bi_iter, _Alloc >, 2413, 2414
format_default
std::regex_constants, 716
format_first_only
std::regex_constants, 716
format_no_copy
std::regex_constants, 716
format_sed
std::regex_constants, 716
formatfwd.h, 3399
formatter.h, 3577
forward
Utilities, 300
forward_as_tuple
Utilities, 301
forward_list, 3681, 3682
std::forward_list< _Tp, _Alloc >, 2142, 2143, 2145
forward_list.h, 3400
forward_list.tcc, 3400
fpos
std::fpos< _StateT >, 2158
frac_digits
std::moneypunct< _CharT, _Intl >, 2456
std::moneypunct_byname< _CharT, _Intl >, 2463
frexp
std, 594
from_bytes
std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >, 3350, 3351
from_chars
std, 594
front
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 836
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1519
std::basic_string< _CharT, _Traits, _Alloc >, 1604
std::chrono::tzdb_list, 3137
std::deque< _Tp, _Alloc >, 2044
std::forward_list< _Tp, _Alloc >, 2149, 2150
std::list< _Tp, _Alloc >, 2337
std::queue< _Tp, _Sequence >, 2777, 2778
std::vector< _Tp, _Alloc >, 3301
std::vector< bool, _Alloc >, 3321
front_insert_iterator
std::front_insert_iterator< _Container >, 2163
front_inserter
Iterators, 171
fs_dir.h, 3401
fs_fwd.h, 3402, 3404
fs_ops.h, 3405, 3408
fs_path.h, 3410
fstream, 3683
I/O, 165
fstream.tcc, 3411
funcref_impl.h, 3411
functexcept.h, 3411
function
std::function< _Res(_ArgTypes...)>, 2165, 2166
Function Objects, 307
mem_fn, 308
function_ref
std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>, 2171
functional, 3684–3686
functional_hash.h, 3412
functions.h, 3577
funcwrap.h, 3413
future, 3688
std::future< _Res >, 2173
std::future< _Res & >, 2175
std::future< void >, 2177
future_category
Futures, 115
future_errc
Futures, 114
future_status
Futures, 114
Futures, 112
async, 114
future_category, 115
future_errc, 114
future_status, 114
launch, 114
make_error_code, 115
make_error_condition, 115
swap, 115

gamma_distribution
std::gamma_distribution< _RealType >, 2180

gbump
__gnu_cxx::enc_filebuf< _CharT >, 2088
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2994
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3013
std::basic_filebuf< _CharT, _Traits >, 1095
std::basic_streambuf< _CharT, _Traits >, 1461
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1654
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3333
gcd
std::experimental, 701
gcount
std::basic_fstream< _CharT, _Traits >, 1124
std::basic_ifstream< _CharT, _Traits >, 1175
std::basic_iostream< _CharT, _Traits >, 1236
std::basic_istream< _CharT, _Traits >, 1282
std::basic_istream< _CharT, _Traits >::sentry, 2874
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1321
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1681
Generalized Numeric operations, 5
accumulate, 6, 7
adjacent_difference, 7, 8
exclusive_scan, 8, 9
inclusive_scan, 9, 10
inner_product, 11
iota, 12
partial_sum, 12, 13
reduce, 13, 14
transform_exclusive_scan, 15
transform_inclusive_scan, 15, 16
transform_reduce, 17, 18
generate
Mutating, 24
generate_canonical
Random Number Generation, 233
generate_minimal_n
__gnu_parallel::_Settings, 1011
generate_n
Mutating, 24
generator, 3689
generic_category
Diagnostics, 124
get
__gnu_parallel::_Settings, 1010
std, 594–596
std::auto_ptr< _Tp >, 1074
std::basic_fstream< _CharT, _Traits >, 1124–1126
std::basic_ifstream< _CharT, _Traits >, 1175–1177
std::basic_iostream< _CharT, _Traits >, 1236, 1237, 1239
std::basic_istream< _CharT, _Traits >, 1282–1284
std::basic_istream< _CharT, _Traits >::sentry, 2874, 2875
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1321–1323
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1681–1683
std::future< _Res >, 2173
std::future< _Res & >, 2175
std::future< void >, 2177
std::money_get< _CharT, _InIter >, 2443, 2444
std::num_get< _CharT, _InIter >, 2554–2560
std::shared_future< _Res >, 2953
std::shared_future< _Res & >, 2955, 2956
std::shared_future< void >, 2958
std::shared_ptr< _Tp >, 2967
std::time_get< _CharT, _InIter >, 3068, 3069
std::time_get_byname< _CharT, _InIter >, 3079
std::unique_ptr< _Tp, _Dp >, 3152
std::unique_ptr< _Tp[], _Dp >, 3160
Utilities, 301
get_actual_size
__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >, 2236
get_allocator
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 836
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1519
std::basic_string< _CharT, _Traits, _Alloc >, 1604
std::deque< _Tp, _Alloc >, 2044
std::forward_list< _Tp, _Alloc >, 2150
std::list< _Tp, _Alloc >, 2337
std::map< _Key, _Tp, _Compare, _Alloc >, 2395
std::match_results< _Bi_iter, _Alloc >, 2414
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2490
std::multiset< _Key, _Compare, _Alloc >, 2517
std::set< _Key, _Compare, _Alloc >, 2944
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2076
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3187
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3216
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3244
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3273
std::vector< _Tp, _Alloc >, 3301
std::vector< bool, _Alloc >, 3321, 3322
get_child
__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 936

__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 938
get_comb_hash_fn
__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >, 1788
get_comb_probe_fn
__gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >, 2194
get_date
std::time_get< _CharT, _InIter >, 3070
std::time_get_byname< _CharT, _InIter >, 3080
get_default_resource
memory_resource, 3707
get_deleter
Pointer Abstractions, 338
std::experimental, 701
std::unique_ptr< _Tp, _Dp >, 3153
std::unique_ptr< _Tp[], _Dp >, 3160
get_eq_fn
__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >, 1789
__gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >, 2195
get_hash_fn
__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >, 1789
__gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >, 2195
get_id
std::this_thread, 721
get_l_child
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1724
__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1729
__gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >, 2699
__gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >, 2701
get_load
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1777
get_loads
__gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >, 2225
__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >, 2236
get_metadata
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1724
__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1729
__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 936
__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 939
get_money
std, 596
get_monthname
std::time_get< _CharT, _InIter >, 3070
std::time_get_byname< _CharT, _InIter >, 3081
get_nearest_larger_size
__gnu_pbds::sample_size_policy, 2854
get_nearest_smaller_size
__gnu_pbds::sample_size_policy, 2854
get_new_handler
std, 596
get_new_size
__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >, 2236
__gnu_pbds::sample_resize_policy, 2849
get_pointer_safety
Pointer Safety and Garbage Collection, 344
get_probe_fn
__gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >, 2195
get_r_child
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1725
__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1729
__gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >, 2699
__gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >, 2701
get_resize_policy
__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >, 1789
__gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy

>, 2195, 2196
get_size_policy
__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >, 2237
get_temporary_buffer
std, 596
get_terminate
Exceptions, 129
get_time
std, 596
std::time_get< _CharT, _InIter >, 3071
std::time_get_byname< _CharT, _InIter >, 3081
get_trigger_policy
__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >, 2237
get_unexpected
Exceptions, 129
get_weekday
std::time_get< _CharT, _InIter >, 3071
std::time_get_byname< _CharT, _InIter >, 3082
get_year
std::time_get< _CharT, _InIter >, 3072
std::time_get_byname< _CharT, _InIter >, 3082
getline
std, 597, 598
std::basic_fstream< _CharT, _Traits >, 1127, 1128
std::basic_ifstream< _CharT, _Traits >, 1178, 1179
std::basic_iostream< _CharT, _Traits >, 1239, 1240
std::basic_istream< _CharT, _Traits >, 1284, 1285
std::basic_istream< _CharT, _Traits >::sentry, 2876, 2878
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1324
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1684, 1685
getloc
__gnu_cxx::enc_filebuf< _CharT >, 2088
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2994
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3013
std::basic_filebuf< _CharT, _Traits >, 1095
std::basic_fstream< _CharT, _Traits >, 1128
std::basic_ifstream< _CharT, _Traits >, 1179
std::basic_ios< _CharT, _Traits >, 1213
std::basic_iostream< _CharT, _Traits >, 1240
std::basic_istream< _CharT, _Traits >, 1285
std::basic_istream< _CharT, _Traits >::sentry, 2878
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1325
std::basic_ofstream< _CharT, _Traits >, 1360
std::basic_ostream< _CharT, _Traits >, 1393
std::basic_ostream< _CharT, _Traits >::sentry, 2907
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1425
std::basic_regex< _Ch_type, _Rx_traits >, 1453
std::basic_streambuf< _CharT, _Traits >, 1461
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1656
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1685
std::ios_base, 2261
std::regex_traits< _Ch_type >, 2825
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3333
global
std::locale, 2354
good
std::basic_fstream< _CharT, _Traits >, 1128
std::basic_ifstream< _CharT, _Traits >, 1179
std::basic_ios< _CharT, _Traits >, 1213
std::basic_iostream< _CharT, _Traits >, 1241
std::basic_istream< _CharT, _Traits >, 1286
std::basic_istream< _CharT, _Traits >::sentry, 2878
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1325
std::basic_ofstream< _CharT, _Traits >, 1360
std::basic_ostream< _CharT, _Traits >, 1393
std::basic_ostream< _CharT, _Traits >::sentry, 2907
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1426
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1685
goodbit
std::basic_fstream< _CharT, _Traits >, 1157
std::basic_ifstream< _CharT, _Traits >, 1200
std::basic_ios< _CharT, _Traits >, 1222
std::basic_iostream< _CharT, _Traits >, 1269
std::basic_istream< _CharT, _Traits >, 1306
std::basic_istream< _CharT, _Traits >::sentry, 2896
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1346
std::basic_ofstream< _CharT, _Traits >, 1380
std::basic_ostream< _CharT, _Traits >, 1410
std::basic_ostream< _CharT, _Traits >::sentry, 2924
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1444
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1714
std::ios_base, 2268
gp_hash_table
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >, 2186–2190
gp_ht_map_.hpp, 3636
gptr
__gnu_cxx::enc_filebuf< _CharT >, 2088
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2994

__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3014
std::basic_filebuf< _CharT, _Traits >, 1095
std::basic_streambuf< _CharT, _Traits >, 1462
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1656
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3333
grep
std::regex_constants, 717
grouping
std::moneypunct< _CharT, _Intl >, 2456
std::moneypunct_byname< _CharT, _Intl >, 2464
std::numpunct< _CharT >, 2677
std::numpunct_byname< _CharT >, 2681
gslice
Numeric Arrays, 218
gslice.h, 3413
gslice_array
Numeric Arrays, 219
gslice_array.h, 3413

has_denorm
std::__numeric_limits_base, 788
std::numeric_limits< _Tp >, 2576
std::numeric_limits< bool >, 2581
std::numeric_limits< char >, 2586
std::numeric_limits< char16_t >, 2591
std::numeric_limits< char32_t >, 2597
std::numeric_limits< double >, 2602
std::numeric_limits< float >, 2607
std::numeric_limits< int >, 2612
std::numeric_limits< long >, 2618
std::numeric_limits< long double >, 2623
std::numeric_limits< long long >, 2628
std::numeric_limits< short >, 2633
std::numeric_limits< signed char >, 2639
std::numeric_limits< unsigned char >, 2644
std::numeric_limits< unsigned int >, 2649
std::numeric_limits< unsigned long >, 2654
std::numeric_limits< unsigned long long >, 2660
std::numeric_limits< unsigned short >, 2665
std::numeric_limits< wchar_t >, 2670
has_denorm_loss
std::__numeric_limits_base, 788
std::numeric_limits< _Tp >, 2576
std::numeric_limits< bool >, 2581
std::numeric_limits< char >, 2586
std::numeric_limits< char16_t >, 2592
std::numeric_limits< char32_t >, 2597
std::numeric_limits< double >, 2602
std::numeric_limits< float >, 2607
std::numeric_limits< int >, 2612
std::numeric_limits< long >, 2618
std::numeric_limits< long double >, 2623
std::numeric_limits< long long >, 2628
std::numeric_limits< short >, 2634
std::numeric_limits< signed char >, 2639
std::numeric_limits< unsigned char >, 2644
std::numeric_limits< unsigned int >, 2649
std::numeric_limits< unsigned long >, 2655
std::numeric_limits< unsigned long long >, 2660
std::numeric_limits< unsigned short >, 2665
std::numeric_limits< wchar_t >, 2670
has_facet
Locales, 174
std::locale, 2356
std::locale::id, 2239
has_infinity
std::__numeric_limits_base, 788
std::numeric_limits< _Tp >, 2576
std::numeric_limits< bool >, 2581
std::numeric_limits< char >, 2586
std::numeric_limits< char16_t >, 2592
std::numeric_limits< char32_t >, 2597
std::numeric_limits< double >, 2602
std::numeric_limits< float >, 2607
std::numeric_limits< int >, 2613
std::numeric_limits< long >, 2618
std::numeric_limits< long double >, 2623
std::numeric_limits< long long >, 2628
std::numeric_limits< short >, 2634
std::numeric_limits< signed char >, 2639
std::numeric_limits< unsigned char >, 2644
std::numeric_limits< unsigned int >, 2649
std::numeric_limits< unsigned long >, 2655
std::numeric_limits< unsigned long long >, 2660
std::numeric_limits< unsigned short >, 2665
std::numeric_limits< wchar_t >, 2670
has_quiet_NaN
std::__numeric_limits_base, 789
std::numeric_limits< _Tp >, 2576
std::numeric_limits< bool >, 2581
std::numeric_limits< char >, 2586
std::numeric_limits< char16_t >, 2592
std::numeric_limits< char32_t >, 2597
std::numeric_limits< double >, 2602
std::numeric_limits< float >, 2607
std::numeric_limits< int >, 2613
std::numeric_limits< long >, 2618
std::numeric_limits< long double >, 2623
std::numeric_limits< long long >, 2628
std::numeric_limits< short >, 2634
std::numeric_limits< signed char >, 2639
std::numeric_limits< unsigned char >, 2644
std::numeric_limits< unsigned int >, 2649
std::numeric_limits< unsigned long >, 2655
std::numeric_limits< unsigned long long >, 2660
std::numeric_limits< unsigned short >, 2665
std::numeric_limits< wchar_t >, 2670

has_signaling_NaN
std::__numeric_limits_base, 789
std::numeric_limits< _Tp >, 2576
std::numeric_limits< bool >, 2581
std::numeric_limits< char >, 2586
std::numeric_limits< char16_t >, 2592
std::numeric_limits< char32_t >, 2597
std::numeric_limits< double >, 2602
std::numeric_limits< float >, 2607
std::numeric_limits< int >, 2613
std::numeric_limits< long >, 2618
std::numeric_limits< long double >, 2623
std::numeric_limits< long long >, 2628
std::numeric_limits< short >, 2634
std::numeric_limits< signed char >, 2639
std::numeric_limits< unsigned char >, 2644
std::numeric_limits< unsigned int >, 2649
std::numeric_limits< unsigned long >, 2655
std::numeric_limits< unsigned long long >, 2660
std::numeric_limits< unsigned short >, 2665
std::numeric_limits< wchar_t >, 2670
hash
std::__cxx11::collate< _CharT >, 1860
Hash-Based, 141
hash_bytes.h, 3413
hash_eq_fn.hpp, 3635
hash_exponential_size_policy
__gnu_pbds::hash_exponential_size_policy< Size_Type >, 2223
hash_exponential_size_policy_imp.hpp, 3647
hash_fun.h, 3361
hash_function
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3187
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3216
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3245
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3273
hash_load_check_resize_trigger
__gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >, 2225
hash_load_check_resize_trigger_imp.hpp, 3647
hash_load_check_resize_trigger_size_base.hpp, 3647
hash_map, 3361
hash_policy.hpp, 3654
hash_prime_size_policy
__gnu_pbds::hash_prime_size_policy, 2232
hash_prime_size_policy_imp.hpp, 3648
hash_set, 3362
hash_standard_resize_policy
__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >, 2236
hash_standard_resize_policy_imp.hpp, 3648
hasher
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3172
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3202
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3231
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3257
Hashes, 315
hashtable.h, 3414
hashtable_policy.h, 3415
Heap, 86
is_heap, 87
is_heap_until, 88
make_heap, 89
pop_heap, 89, 90
push_heap, 90, 91
sort_heap, 91
Heap-Based, 142
helper_functions.h, 3579
hermite
Mathematical Special Functions, 208
TR1 Mathematical Special Functions, 250
hermitef
Mathematical Special Functions, 208
hermitel
Mathematical Special Functions, 209
hex
std, 599
std::basic_fstream< _CharT, _Traits >, 1158
std::basic_ifstream< _CharT, _Traits >, 1200
std::basic_ios< _CharT, _Traits >, 1223
std::basic_iostream< _CharT, _Traits >, 1269
std::basic_istream< _CharT, _Traits >, 1307
std::basic_istream< _CharT, _Traits >::sentry, 2896
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1346
std::basic_ofstream< _CharT, _Traits >, 1380
std::basic_ostream< _CharT, _Traits >, 1410
std::basic_ostream< _CharT, _Traits >::sentry, 2924
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1444
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1715
std::ios_base, 2268
hexfloat
std, 599
high_resolution_clock
Time, 365
hours
Time, 365

hyperg
Mathematical Special Functions, 209
std::tr1, 725
hypergf
Mathematical Special Functions, 209
std::tr1, 726
hypergl
Mathematical Special Functions, 210
std::tr1, 726

I/O, 163
filebuf, 165
fstream, 165
ifstream, 165
ios, 165
iostream, 165
istream, 165
istringstream, 165
ofstream, 165
ostream, 165
ostringstream, 165
streambuf, 165
stringbuf, 165
stringstream, 166
wfilebuf, 166
wfstream, 166
wifstream, 166
wios, 166
wiostream, 166
wistream, 166
wistringstream, 166
wofstream, 166
wostream, 166
wostringstream, 166
wstreambuf, 167
wstringbuf, 167
wstringstream, 167
icase
std::regex_constants, 717
id
std::__cxx11::collate< _CharT >, 1861
std::ctype< _CharT >, 1923
std::ctype< char >, 1958
std::ctype< wchar_t >, 1976
std::ctype_byname< _CharT >, 1990
std::ctype_byname< char >, 2013
std::locale::id, 2239
std::messages< _CharT >, 2432
std::messages_byname< _CharT >, 2434
std::money_get< _CharT, _InIter >, 2445
std::money_put< _CharT, _OutIter >, 2449
std::moneypunct< _CharT, _Intl >, 2458
std::moneypunct_byname< _CharT, _Intl >, 2466
std::num_get< _CharT, _InIter >, 2560
std::num_put< _CharT, _OutIter >, 2573
std::numpunct< _CharT >, 2678
std::numpunct_byname< _CharT >, 2682
std::time_get< _CharT, _InIter >, 3073
std::time_get_byname< _CharT, _InIter >, 3083
std::time_put< _CharT, _OutIter >, 3088
std::time_put_byname< _CharT, _OutIter >, 3091
identity_element
SGI, 154
ifstream
I/O, 165
ignore
std, 646
std::basic_fstream< _CharT, _Traits >, 1128, 1129
std::basic_ifstream< _CharT, _Traits >, 1179, 1180
std::basic_iostream< _CharT, _Traits >, 1241
std::basic_istream< _CharT, _Traits >, 1286
std::basic_istream< _CharT, _Traits >::sentry, 2878, 2879
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1325
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1685, 1686
imbue
__gnu_cxx::enc_filebuf< _CharT >, 2088
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2995
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3014
std::basic_filebuf< _CharT, _Traits >, 1096
std::basic_fstream< _CharT, _Traits >, 1129
std::basic_ifstream< _CharT, _Traits >, 1180
std::basic_ios< _CharT, _Traits >, 1213
std::basic_iostream< _CharT, _Traits >, 1243
std::basic_istream< _CharT, _Traits >, 1288
std::basic_istream< _CharT, _Traits >::sentry, 2879
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1326
std::basic_ofstream< _CharT, _Traits >, 1361
std::basic_ostream< _CharT, _Traits >, 1393
std::basic_ostream< _CharT, _Traits >::sentry, 2908
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1426
std::basic_regex< _Ch_type, _Rx_traits >, 1453
std::basic_streambuf< _CharT, _Traits >, 1462
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1656
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1686
std::ios_base, 2262
std::regex_traits< _Ch_type >, 2825
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3334
in
std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >, 744
std::basic_fstream< _CharT, _Traits >, 1158

std::basic_ifstream< _CharT, _Traits >, 1200
std::basic_ios< _CharT, _Traits >, 1223
std::basic_iostream< _CharT, _Traits >, 1269
std::basic_istream< _CharT, _Traits >, 1307
std::basic_istream< _CharT, _Traits >::sentry, 2896
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1346
std::basic_ofstream< _CharT, _Traits >, 1380
std::basic_ostream< _CharT, _Traits >, 1410
std::basic_ostream< _CharT, _Traits >::sentry, 2924
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1444
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1715
std::codecvt< _InternT, _ExternT, _StateT >, 1803
std::codecvt< _InternT, _ExternT, encoding_state >, 1809, 1810
std::codecvt< char, char, mbstate_t >, 1818, 1819
std::codecvt< char16_t, char, mbstate_t >, 1827, 1828
std::codecvt< char32_t, char, mbstate_t >, 1836, 1837
std::codecvt< wchar_t, char, mbstate_t >, 1845, 1846
std::codecvt_byname< _InternT, _ExternT, _StateT >, 1854
std::ios_base, 2268
in_avail
__gnu_cxx::enc_filebuf< _CharT >, 2089
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2995
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3014
std::basic_filebuf< _CharT, _Traits >, 1096
std::basic_streambuf< _CharT, _Traits >, 1462
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1657
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3334
in_place
Optional values, 287
includes
Set Operations, 93
inclusive_scan
Generalized Numeric operations, 9, 10
increment
std::linear_congruential_engine< _UIntType, __a, __c, __m >, 2323
independent_bits_engine
std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >, 2242, 2243
index_sequence
std, 579
index_sequence_for
std, 579
indirect.h, 3415
indirect_array
Numeric Arrays, 219
indirect_array.h, 3415
infinity
std::numeric_limits< _Tp >, 2574
std::numeric_limits< bool >, 2580
std::numeric_limits< char >, 2585
std::numeric_limits< char16_t >, 2590
std::numeric_limits< char32_t >, 2596
std::numeric_limits< double >, 2601
std::numeric_limits< float >, 2606
std::numeric_limits< int >, 2611
std::numeric_limits< long >, 2617
std::numeric_limits< long double >, 2622
std::numeric_limits< long long >, 2627
std::numeric_limits< short >, 2632
std::numeric_limits< signed char >, 2638
std::numeric_limits< unsigned char >, 2643
std::numeric_limits< unsigned int >, 2648
std::numeric_limits< unsigned long >, 2653
std::numeric_limits< unsigned long long >, 2659
std::numeric_limits< unsigned short >, 2664
std::numeric_limits< wchar_t >, 2669
info_fn_imps.hpp, 3622, 3623
init
std::basic_fstream< _CharT, _Traits >, 1130
std::basic_ifstream< _CharT, _Traits >, 1181
std::basic_ios< _CharT, _Traits >, 1213
std::basic_iostream< _CharT, _Traits >, 1243
std::basic_istream< _CharT, _Traits >, 1288
std::basic_istream< _CharT, _Traits >::sentry, 2880
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1326
std::basic_ofstream< _CharT, _Traits >, 1361
std::basic_ostream< _CharT, _Traits >, 1394
std::basic_ostream< _CharT, _Traits >::sentry, 2908
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1426
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1687
initializer_list, 3805
inner_product
Generalized Numeric operations, 11
inplace_merge
Sorting, 63, 64
inplace_vector, 3689
insert
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 836–840
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1519–1525
std::basic_string< _CharT, _Traits, _Alloc >, 1604–1611, 1613, 1614
std::deque< _Tp, _Alloc >, 2044–2046
std::list< _Tp, _Alloc >, 2338, 2339

std::map< _Key, _Tp, _Compare, _Alloc >, 2395–2398
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2490–2493
std::multiset< _Key, _Compare, _Alloc >, 2517–2519
std::set< _Key, _Compare, _Alloc >, 2945, 2946
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3187–3190
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3216, 3217, 3219
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3245–3247
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3273–3275
std::vector< _Tp, _Alloc >, 3301–3303
std::vector< bool, _Alloc >, 3322, 3323
insert_after
std::forward_list< _Tp, _Alloc >, 2150, 2151
insert_fn_imps.hpp, 3623, 3624
insert_iterator
std::insert_iterator< _Container >, 2253
insert_join_fn_imps.hpp, 3643
insert_no_store_hash_fn_imps.hpp, 3633
insert_or_assign
std::map< _Key, _Tp, _Compare, _Alloc >, 2399
insert_store_hash_fn_imps.hpp, 3633
inserter
Iterators, 171
int_type
std::basic_ios< _CharT, _Traits >, 1207
std::basic_streambuf< _CharT, _Traits >, 1459
std::istreambuf_iterator< _CharT, _Traits >, 2294
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3331
internal
std, 599
std::basic_fstream< _CharT, _Traits >, 1158
std::basic_ifstream< _CharT, _Traits >, 1200
std::basic_ios< _CharT, _Traits >, 1223
std::basic_iostream< _CharT, _Traits >, 1269
std::basic_istream< _CharT, _Traits >, 1307
std::basic_istream< _CharT, _Traits >::sentry, 2896
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1346
std::basic_ofstream< _CharT, _Traits >, 1381
std::basic_ostream< _CharT, _Traits >, 1411
std::basic_ostream< _CharT, _Traits >::sentry, 2924
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1444
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1715
std::ios_base, 2268
intervals
std::piecewise_constant_distribution< _RealType >, 2744
std::piecewise_linear_distribution< _RealType >, 2748
intl
std::moneypunct< _CharT, _Intl >, 2458
Invalidation Guarantees, 145
invoke.h, 3415
io_errc
std, 581
iomanip, 3689
ios, 3691
I/O, 165
ios_base.h, 3416
iosfwd, 3691
iostate
std::basic_fstream< _CharT, _Traits >, 1117
std::basic_ifstream< _CharT, _Traits >, 1168
std::basic_ios< _CharT, _Traits >, 1207
std::basic_iostream< _CharT, _Traits >, 1231
std::basic_istream< _CharT, _Traits >, 1277
std::basic_istream< _CharT, _Traits >::sentry, 2869
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1316
std::basic_ofstream< _CharT, _Traits >, 1353
std::basic_ostream< _CharT, _Traits >, 1388
std::basic_ostream< _CharT, _Traits >::sentry, 2902
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1419
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1676
std::ios_base, 2260
iostream, 3692
I/O, 165
iota
__gnu_cxx, 394
Generalized Numeric operations, 12
is
std::__ctype_abstract_base< _CharT >, 757
std::ctype< _CharT >, 1917, 1918
std::ctype< char >, 1942–1944
std::ctype< wchar_t >, 1971, 1972
std::ctype_byname< _CharT >, 1985
std::ctype_byname< char >, 2003, 2004
is_always_equal
__gnu_cxx::__alloc_traits< _Alloc, typename >, 734
std::allocator_traits< _Alloc >, 1033
std::allocator_traits< allocator< _Tp > >, 1039
std::allocator_traits< allocator< void > >, 1047, 1048
std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1056
is_bind_expression_v
std::experimental, 703
is_bounded

std::__numeric_limits_base, 789
std::numeric_limits< _Tp >, 2576
std::numeric_limits< bool >, 2581
std::numeric_limits< char >, 2587
std::numeric_limits< char16_t >, 2592
std::numeric_limits< char32_t >, 2597
std::numeric_limits< double >, 2602
std::numeric_limits< float >, 2608
std::numeric_limits< int >, 2613
std::numeric_limits< long >, 2618
std::numeric_limits< long double >, 2623
std::numeric_limits< long long >, 2629
std::numeric_limits< short >, 2634
std::numeric_limits< signed char >, 2639
std::numeric_limits< unsigned char >, 2644
std::numeric_limits< unsigned int >, 2650
std::numeric_limits< unsigned long >, 2655
std::numeric_limits< unsigned long long >, 2660
std::numeric_limits< unsigned short >, 2665
std::numeric_limits< wchar_t >, 2671
is_corresponding_member
Metaprogramming, 352
is_detected
Detection idiom, 285
is_detected_convertible
Detection idiom, 285
is_detected_convertible_v
Detection idiom, 285
is_detected_exact
Detection idiom, 285
is_detected_exact_v
Detection idiom, 286
is_detected_v
Detection idiom, 286
is_emptyset
std::tr2::bool_set, 1771
is_exact
std::__numeric_limits_base, 789
std::numeric_limits< _Tp >, 2576
std::numeric_limits< bool >, 2581
std::numeric_limits< char >, 2587
std::numeric_limits< char16_t >, 2592
std::numeric_limits< char32_t >, 2597
std::numeric_limits< double >, 2602
std::numeric_limits< float >, 2608
std::numeric_limits< int >, 2613
std::numeric_limits< long >, 2618
std::numeric_limits< long double >, 2623
std::numeric_limits< long long >, 2629
std::numeric_limits< short >, 2634
std::numeric_limits< signed char >, 2639
std::numeric_limits< unsigned char >, 2644
std::numeric_limits< unsigned int >, 2650
std::numeric_limits< unsigned long >, 2655
std::numeric_limits< unsigned long long >, 2660
std::numeric_limits< unsigned short >, 2665
std::numeric_limits< wchar_t >, 2671
is_grow_needed
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1777
__gnu_pbds::sample_resize_trigger, 2852
is_heap
Heap, 87
is_heap_until
Heap, 88
is_iec559
std::__numeric_limits_base, 789
std::numeric_limits< _Tp >, 2576
std::numeric_limits< bool >, 2582
std::numeric_limits< char >, 2587
std::numeric_limits< char16_t >, 2592
std::numeric_limits< char32_t >, 2597
std::numeric_limits< double >, 2603
std::numeric_limits< float >, 2608
std::numeric_limits< int >, 2613
std::numeric_limits< long >, 2618
std::numeric_limits< long double >, 2624
std::numeric_limits< long long >, 2629
std::numeric_limits< short >, 2634
std::numeric_limits< signed char >, 2639
std::numeric_limits< unsigned char >, 2645
std::numeric_limits< unsigned int >, 2650
std::numeric_limits< unsigned long >, 2655
std::numeric_limits< unsigned long long >, 2660
std::numeric_limits< unsigned short >, 2666
std::numeric_limits< wchar_t >, 2671
is_indeterminate
std::tr2::bool_set, 1771
is_integer
std::__numeric_limits_base, 789
std::numeric_limits< _Tp >, 2576
std::numeric_limits< bool >, 2582
std::numeric_limits< char >, 2587
std::numeric_limits< char16_t >, 2592
std::numeric_limits< char32_t >, 2597
std::numeric_limits< double >, 2603
std::numeric_limits< float >, 2608
std::numeric_limits< int >, 2613
std::numeric_limits< long >, 2618
std::numeric_limits< long double >, 2624
std::numeric_limits< long long >, 2629
std::numeric_limits< short >, 2634
std::numeric_limits< signed char >, 2639
std::numeric_limits< unsigned char >, 2645
std::numeric_limits< unsigned int >, 2650
std::numeric_limits< unsigned long >, 2655
std::numeric_limits< unsigned long long >, 2660
std::numeric_limits< unsigned short >, 2666

std::numeric_limits< wchar_t >, 2671
is_layout_compatible_v
Metaprogramming, 353
is_modulo
std::__numeric_limits_base, 789
std::numeric_limits< _Tp >, 2577
std::numeric_limits< bool >, 2582
std::numeric_limits< char >, 2587
std::numeric_limits< char16_t >, 2592
std::numeric_limits< char32_t >, 2598
std::numeric_limits< double >, 2603
std::numeric_limits< float >, 2608
std::numeric_limits< int >, 2613
std::numeric_limits< long >, 2618
std::numeric_limits< long double >, 2624
std::numeric_limits< long long >, 2629
std::numeric_limits< short >, 2634
std::numeric_limits< signed char >, 2640
std::numeric_limits< unsigned char >, 2645
std::numeric_limits< unsigned int >, 2650
std::numeric_limits< unsigned long >, 2655
std::numeric_limits< unsigned long long >, 2661
std::numeric_limits< unsigned short >, 2666
std::numeric_limits< wchar_t >, 2671
is_open
__gnu_cxx::enc_filebuf< _CharT >, 2089
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2995
std::basic_filebuf< _CharT, _Traits >, 1096
std::basic_fstream< _CharT, _Traits >, 1130
std::basic_ifstream< _CharT, _Traits >, 1181
std::basic_ofstream< _CharT, _Traits >, 1361
is_partitioned
Mutating, 25
is_permutation
Non-Mutating, 53, 54
is_placeholder_v
std::experimental, 703
is_pointer_interconvertible_base_of_v
Metaprogramming, 354
is_pointer_interconvertible_with_class
Metaprogramming, 353
is_resize_needed
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< Comb_Probe_Fn, Hash_Fn, Allocator, External_Load_Access, Size_Type >, 1777
__gnu_pbds::sample_resize_policy, 2849
__gnu_pbds::sample_resize_trigger, 2852
is_signed
std::__numeric_limits_base, 789
std::numeric_limits< _Tp >, 2577
std::numeric_limits< bool >, 2582
std::numeric_limits< char >, 2587
std::numeric_limits< char16_t >, 2592
std::numeric_limits< char32_t >, 2598
std::numeric_limits< double >, 2603
std::numeric_limits< float >, 2608
std::numeric_limits< int >, 2613
std::numeric_limits< long >, 2619
std::numeric_limits< long double >, 2624
std::numeric_limits< long long >, 2629
std::numeric_limits< short >, 2634
std::numeric_limits< signed char >, 2640
std::numeric_limits< unsigned char >, 2645
std::numeric_limits< unsigned int >, 2650
std::numeric_limits< unsigned long >, 2655
std::numeric_limits< unsigned long long >, 2661
std::numeric_limits< unsigned short >, 2666
std::numeric_limits< wchar_t >, 2671
is_singleton
std::tr2::bool_set, 1771
is_sorted
Sorting, 64, 65
is_sorted_until
Sorting, 65
is_specialized
std::__numeric_limits_base, 789
std::numeric_limits< _Tp >, 2577
std::numeric_limits< bool >, 2582
std::numeric_limits< char >, 2587
std::numeric_limits< char16_t >, 2593
std::numeric_limits< char32_t >, 2598
std::numeric_limits< double >, 2603
std::numeric_limits< float >, 2608
std::numeric_limits< int >, 2613
std::numeric_limits< long >, 2619
std::numeric_limits< long double >, 2624
std::numeric_limits< long long >, 2629
std::numeric_limits< short >, 2635
std::numeric_limits< signed char >, 2640
std::numeric_limits< unsigned char >, 2645
std::numeric_limits< unsigned int >, 2650
std::numeric_limits< unsigned long >, 2656
std::numeric_limits< unsigned long long >, 2661
std::numeric_limits< unsigned short >, 2666
std::numeric_limits< wchar_t >, 2671
isalnum
std, 599
isalpha
std, 599
isblank
std, 600
iscntrl
std, 600
isctype
std::regex_traits< _Ch_type >, 2825
isdigit
std, 600
isgraph
std, 600

islower
std, 600
isprint
std, 600
ispunct
std, 601
isspace
std, 601
istream, 3693
I/O, 165
istream.tcc, 3417
istream_iterator
std::istream_iterator< _Tp, _CharT, _Traits, _Dist >, 2292
istream_type
std::istreambuf_iterator< _CharT, _Traits >, 2294
istreambuf_iterator
std::istreambuf_iterator< _CharT, _Traits >, 2295, 2296
istringstream
I/O, 165
isupper
std, 601
isxdigit
std, 601
iter_swap
Mutating, 25
iter_type
std::money_get< _CharT, _InIter >, 2442
std::money_put< _CharT, _OutIter >, 2446
std::num_get< _CharT, _InIter >, 2547
std::num_put< _CharT, _OutIter >, 2562
std::time_get< _CharT, _InIter >, 3064
std::time_put< _CharT, _OutIter >, 3086
iterator, 3694
std::set< _Key, _Compare, _Alloc >, 2933
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3172
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3202
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3231
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3257
Iterator Tags, 173
iterator.h, 3732
iterator.hpp, 3653
iterator_category
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1724
__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1728
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1738
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1742
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >, 2306
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_ite
Node, _Alloc >, 2310
std::back_insert_iterator< _Container >, 1077
std::front_insert_iterator< _Container >, 2162
std::insert_iterator< _Container >, 2253
std::istream_iterator< _Tp, _CharT, _Traits, _Dist >, 2292
std::istreambuf_iterator< _CharT, _Traits >, 2295
std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >, 2300
std::ostream_iterator< _Tp, _CharT, _Traits >, 2688
std::ostreambuf_iterator< _CharT, _Traits >, 2691
std::raw_storage_iterator< _OutputIterator, _Tp >, 2796
iterator_concepts.h, 3418
iterator_fn_imps.hpp, 3636
Iterators, 167
__iterator_category, 170
back_inserter, 171
front_inserter, 171
inserter, 171
make_reverse_iterator, 172
operator==, 172
iterators_fn_imps.hpp, 3624, 3625
iword
std::basic_fstream< _CharT, _Traits >, 1130
std::basic_ifstream< _CharT, _Traits >, 1181
std::basic_ios< _CharT, _Traits >, 1214
std::basic_iostream< _CharT, _Traits >, 1243
std::basic_istream< _CharT, _Traits >, 1288
std::basic_istream< _CharT, _Traits >::sentry, 2880
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1326
std::basic_ofstream< _CharT, _Traits >, 1361
std::basic_ostream< _CharT, _Traits >, 1394
std::basic_ostream< _CharT, _Traits >::sentry, 2908
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1426
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1687
std::ios_base, 2262

k
std::negative_binomial_distribution< _IntType >, 2531
key_comp
std::map< _Key, _Tp, _Compare, _Alloc >, 2400
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2494
std::multiset< _Key, _Compare, _Alloc >, 2519

std::set< _Key, _Compare, _Alloc >, 2946
key_compare
std::set< _Key, _Compare, _Alloc >, 2933
key_eq
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3190
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3220
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3247
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3275
key_equal
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3172
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3202
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3231
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3257
key_type
std::set< _Key, _Compare, _Alloc >, 2933
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3172
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3203
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3231
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3257
kill_dependency
Atomics, 110

L1_cache_size
__gnu_parallel::_Settings, 1012
L2_cache_size
__gnu_parallel::_Settings, 1012
laguerre
Mathematical Special Functions, 210
TR1 Mathematical Special Functions, 250
laguerref
Mathematical Special Functions, 210
laguerrel
Mathematical Special Functions, 211
lambda
std::exponential_distribution< _RealType >, 2120
latch, 3695
launch
Futures, 114
lcm
std::experimental, 701
left
std, 601
std::basic_fstream< _CharT, _Traits >, 1158
std::basic_ifstream< _CharT, _Traits >, 1200
std::basic_ios< _CharT, _Traits >, 1223
std::basic_iostream< _CharT, _Traits >, 1270
std::basic_istream< _CharT, _Traits >, 1307
std::basic_istream< _CharT, _Traits >::sentry, 2896
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1346
std::basic_ofstream< _CharT, _Traits >, 1381
std::basic_ostream< _CharT, _Traits >, 1411
std::basic_ostream< _CharT, _Traits >::sentry, 2924
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1444
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1715
std::ios_base, 2269
left_child_next_sibling_heap_.hpp, 3640
left_child_next_sibling_heap_const_iterator_
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >, 2306
left_child_next_sibling_heap_node_point_const_iterator_
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_ite
Node, _Alloc >, 2310
legendre
Mathematical Special Functions, 211
TR1 Mathematical Special Functions, 251
legendref
Mathematical Special Functions, 211
legendrel
Mathematical Special Functions, 212
length
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 841
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1526
std::basic_string< _CharT, _Traits, _Alloc >, 1614, 1615
std::match_results< _Bi_iter, _Alloc >, 2414
std::regex_traits< _Ch_type >, 2826
std::sub_match< _BiIter >, 3037
lexicographical_compare
Sorting, 66
lexicographical_compare_3way
SGI, 155
lexicographical_compare_three_way
Sorting, 67
lfts_config.h, 3603
Library Fundamentals TS, 280
limits, 3695
linear_congruential_engine
std::linear_congruential_engine< _UIntType, __a, __c, __m >, 2319
linear_probe_fn_imp.hpp, 3637
list, 3696, 3697

std::list< _Tp, _Alloc >, 2330–2332
List-Based, 143
list.tcc, 3418
list_partition
__gnu_parallel, 452
list_partition.h, 3733
list_update
__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >, 2348
list_update_policy.hpp, 3655
load_factor
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3190
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3220
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3247
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3275
local_iterator
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3172
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3203
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3231
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3257
local_time_format
Time, 369
locale, 3698
std::locale, 2351–2353
locale_classes.h, 3419
locale_classes.tcc, 3419
locale_conv.h, 3419
locale_facets.h, 3420
locale_facets.tcc, 3422
locale_facets_nonio.h, 3422
locale_facets_nonio.tcc, 3423
localefwd.h, 3424
Locales, 173
has_facet, 174
use_facet, 175
lock
Mutexes, 117
log
Complex Numbers, 181
math.h, 3702
log10
Complex Numbers, 182
math.h, 3702
logic_error
std::logic_error, 2359
Logical operator traits, 286
lookup_classname
std::regex_traits< _Ch_type >, 2826
lookup_collatename
std::regex_traits< _Ch_type >, 2827
losertree.h, 3733
lower_bound
Binary Search, 84
std::map< _Key, _Tp, _Compare, _Alloc >, 2400, 2401
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2494, 2495
std::multiset< _Key, _Compare, _Alloc >, 2519, 2520
std::set< _Key, _Compare, _Alloc >, 2946, 2947
lowest
std::numeric_limits< _Tp >, 2574
std::numeric_limits< bool >, 2580
std::numeric_limits< char >, 2585
std::numeric_limits< char16_t >, 2591
std::numeric_limits< char32_t >, 2596
std::numeric_limits< double >, 2601
std::numeric_limits< float >, 2606
std::numeric_limits< int >, 2612
std::numeric_limits< long >, 2617
std::numeric_limits< long double >, 2622
std::numeric_limits< long long >, 2627
std::numeric_limits< short >, 2633
std::numeric_limits< signed char >, 2638
std::numeric_limits< unsigned char >, 2643
std::numeric_limits< unsigned int >, 2648
std::numeric_limits< unsigned long >, 2654
std::numeric_limits< unsigned long long >, 2659
std::numeric_limits< unsigned short >, 2664
std::numeric_limits< wchar_t >, 2669
lu_counter_metadata.hpp, 3642
lu_map_.hpp, 3641

macros.h, 3581
_GLIBCXX_DEBUG_VERIFY_AT_F, 3584
__glibcxx_check_erase, 3582
__glibcxx_check_erase_after, 3582
__glibcxx_check_erase_range, 3582
__glibcxx_check_erase_range_after, 3582
__glibcxx_check_heap_pred, 3582
__glibcxx_check_insert, 3582
__glibcxx_check_insert_after, 3582
__glibcxx_check_insert_range, 3583
__glibcxx_check_insert_range_after, 3583
__glibcxx_check_partitioned_lower, 3583
__glibcxx_check_partitioned_lower_pred, 3583
__glibcxx_check_partitioned_upper_pred, 3583
__glibcxx_check_sorted_pred, 3584
make_array
Array creation functions, 282
make_boyer_moore_horspool_searcher

std::experimental, 701
make_boyer_moore_searcher
std::experimental, 702
make_default_searcher
std::experimental, 702
make_error_code
Diagnostics, 124
Futures, 115
make_error_condition
Diagnostics, 124
Futures, 115
make_exception_ptr
Exceptions, 130
make_heap
Heap, 89
make_index_sequence
std, 579
make_integer_sequence
std, 579
make_ostream_joiner
std::experimental, 702
make_pair
Utilities, 301
make_reverse_iterator
Iterators, 172
make_shared
Pointer Abstractions, 338
make_signed_t
Metaprogramming, 351
make_tuple
Utilities, 302
make_unsigned_t
Metaprogramming, 351
malloc_allocator.h, 3610
map, 3698, 3699
std::map< _Key, _Tp, _Compare, _Alloc >, 2383–2386
map.h, 3584
mapped_type
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3172
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3203
mark_count
std::basic_regex< _Ch_type, _Rx_traits >, 1453
mask_array
Numeric Arrays, 219
mask_array.h, 3424
mask_based_range_hashing.hpp, 3637
match_any
std::regex_constants, 717
match_continuous
std::regex_constants, 717
match_default
std::regex_constants, 717
match_flag_type
std::regex_constants, 710
match_not_bol
std::regex_constants, 717
match_not_bow
std::regex_constants, 717
match_not_eol
std::regex_constants, 717
match_not_eow
std::regex_constants, 718
match_not_null
std::regex_constants, 718
match_prev_avail
std::regex_constants, 718
match_results
std::match_results< _Bi_iter, _Alloc >, 2411, 2412
math.h, 3700
abs, 3701
cos, 3701
cosh, 3702
exp, 3702
fabs, 3702
log, 3702
log10, 3702
pow, 3702
sin, 3702
sinh, 3702
sqrt, 3703
tan, 3703
tanh, 3703
Mathematical Special Functions, 188
airy_ai, 192
airy_aif, 192
airy_ail, 192
airy_bi, 192
airy_bif, 192
airy_bil, 193
assoc_laguerre, 193
assoc_laguerref, 194
assoc_laguerrel, 194
assoc_legendre, 194
assoc_legendref, 195
assoc_legendrel, 195
beta, 195
betaf, 196
betal, 196
comp_ellint_1, 196
comp_ellint_1f, 197
comp_ellint_1l, 197
comp_ellint_2, 197
comp_ellint_2f, 198
comp_ellint_2l, 198
comp_ellint_3, 198

comp_ellint_3f, 199
comp_ellint_3l, 199
conf_hyperg, 199
conf_hypergf, 200
conf_hypergl, 200
cyl_bessel_i, 200
cyl_bessel_if, 201
cyl_bessel_il, 201
cyl_bessel_j, 201
cyl_bessel_jf, 201
cyl_bessel_jl, 202
cyl_bessel_k, 202
cyl_bessel_kf, 202
cyl_bessel_kl, 203
cyl_neumann, 203
cyl_neumannf, 203
cyl_neumannl, 204
ellint_1, 204
ellint_1f, 204
ellint_1l, 205
ellint_2, 205
ellint_2f, 206
ellint_2l, 206
ellint_3, 206
ellint_3f, 207
ellint_3l, 207
expint, 207
expintf, 208
expintl, 208
hermite, 208
hermitef, 208
hermitel, 209
hyperg, 209
hypergf, 209
hypergl, 210
laguerre, 210
laguerref, 210
laguerrel, 211
legendre, 211
legendref, 211
legendrel, 212
riemann_zeta, 212
riemann_zetaf, 212
riemann_zetal, 213
sph_bessel, 213
sph_besself, 213
sph_bessell, 214
sph_legendre, 214
sph_legendref, 214
sph_legendrel, 215
sph_neumann, 215
sph_neumannf, 215
sph_neumannl, 215
max
__gnu_parallel, 452
Numeric Arrays, 223
Sorting, 67, 68
std::bernoulli_distribution, 1718
std::binomial_distribution< _IntType >, 1749
std::cauchy_distribution< _RealType >, 1775
std::chi_squared_distribution< _RealType >, 1798
std::discard_block_engine< _RandomNumberEngine, __p, __r >, 2059
std::discrete_distribution< _IntType >, 2062
std::exponential_distribution< _RealType >, 2120
std::extreme_value_distribution< _RealType >, 2124
std::fisher_f_distribution< _RealType >, 2133
std::gamma_distribution< _RealType >, 2180
std::geometric_distribution< _IntType >, 2183
std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >, 2243
std::linear_congruential_engine< _UIntType, __a, __c, __m >, 2319
std::lognormal_distribution< _RealType >, 2367
std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >, 2427
std::negative_binomial_distribution< _IntType >, 2531
std::normal_distribution< _RealType >, 2537
std::numeric_limits< _Tp >, 2575
std::numeric_limits< bool >, 2580
std::numeric_limits< char >, 2585
std::numeric_limits< char16_t >, 2591
std::numeric_limits< char32_t >, 2596
std::numeric_limits< double >, 2601
std::numeric_limits< float >, 2606
std::numeric_limits< int >, 2612
std::numeric_limits< long >, 2617
std::numeric_limits< long double >, 2622
std::numeric_limits< long long >, 2627
std::numeric_limits< short >, 2633
std::numeric_limits< signed char >, 2638
std::numeric_limits< unsigned char >, 2643
std::numeric_limits< unsigned int >, 2648
std::numeric_limits< unsigned long >, 2654
std::numeric_limits< unsigned long long >, 2659
std::numeric_limits< unsigned short >, 2664
std::numeric_limits< wchar_t >, 2669
std::piecewise_constant_distribution< _RealType >, 2744
std::piecewise_linear_distribution< _RealType >, 2748
std::poisson_distribution< _IntType >, 2760
std::shuffle_order_engine< _RandomNumberEngine, __k >, 2971
std::student_t_distribution< _RealType >, 3032

std::subtract_with_carry_engine< _UIntType, __w, __s, __r >, 3041
std::uniform_int_distribution< _IntType >, 3144
std::uniform_real_distribution< _RealType >, 3147
std::weibull_distribution< _RealType >, 3347
max_blocks_per_chunk
std::pmr::pool_options, 2764
max_bucket_count
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3191
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3220
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3247
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3276
max_count
__gnu_pbds::lu_counter_policy< Max_Count, _Alloc >, 2371
max_digits10
std::__numeric_limits_base, 789
std::numeric_limits< _Tp >, 2577
std::numeric_limits< bool >, 2582
std::numeric_limits< char >, 2587
std::numeric_limits< char16_t >, 2593
std::numeric_limits< char32_t >, 2598
std::numeric_limits< double >, 2603
std::numeric_limits< float >, 2608
std::numeric_limits< int >, 2614
std::numeric_limits< long >, 2619
std::numeric_limits< long double >, 2624
std::numeric_limits< long long >, 2629
std::numeric_limits< short >, 2635
std::numeric_limits< signed char >, 2640
std::numeric_limits< unsigned char >, 2645
std::numeric_limits< unsigned int >, 2650
std::numeric_limits< unsigned long >, 2656
std::numeric_limits< unsigned long long >, 2661
std::numeric_limits< unsigned short >, 2666
std::numeric_limits< wchar_t >, 2671
max_element
Sorting, 68, 69
max_element_minimal_n
__gnu_parallel::_Settings, 1012
max_exponent
std::__numeric_limits_base, 790
std::numeric_limits< _Tp >, 2577
std::numeric_limits< bool >, 2582
std::numeric_limits< char >, 2587
std::numeric_limits< char16_t >, 2593
std::numeric_limits< char32_t >, 2598
std::numeric_limits< double >, 2603
std::numeric_limits< float >, 2608
std::numeric_limits< int >, 2614
std::numeric_limits< long >, 2619
std::numeric_limits< long double >, 2624
std::numeric_limits< long long >, 2629
std::numeric_limits< short >, 2635
std::numeric_limits< signed char >, 2640
std::numeric_limits< unsigned char >, 2645
std::numeric_limits< unsigned int >, 2650
std::numeric_limits< unsigned long >, 2656
std::numeric_limits< unsigned long long >, 2661
std::numeric_limits< unsigned short >, 2666
std::numeric_limits< wchar_t >, 2671
max_exponent10
std::__numeric_limits_base, 790
std::numeric_limits< _Tp >, 2577
std::numeric_limits< bool >, 2582
std::numeric_limits< char >, 2587
std::numeric_limits< char16_t >, 2593
std::numeric_limits< char32_t >, 2598
std::numeric_limits< double >, 2603
std::numeric_limits< float >, 2608
std::numeric_limits< int >, 2614
std::numeric_limits< long >, 2619
std::numeric_limits< long double >, 2624
std::numeric_limits< long long >, 2629
std::numeric_limits< short >, 2635
std::numeric_limits< signed char >, 2640
std::numeric_limits< unsigned char >, 2645
std::numeric_limits< unsigned int >, 2650
std::numeric_limits< unsigned long >, 2656
std::numeric_limits< unsigned long long >, 2661
std::numeric_limits< unsigned short >, 2666
std::numeric_limits< wchar_t >, 2671
max_load_factor
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3191
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3220
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3247, 3248
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3276
max_size
__gnu_cxx::__alloc_traits< _Alloc, typename >, 738
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 841
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1526
std::allocator_traits< _Alloc >, 1036
std::allocator_traits< allocator< _Tp > >, 1044
std::allocator_traits< allocator< void > >, 1053
std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1061, 1062
std::basic_string< _CharT, _Traits, _Alloc >, 1615
std::deque< _Tp, _Alloc >, 2046

std::forward_list< _Tp, _Alloc >, 2151
std::list< _Tp, _Alloc >, 2340
std::map< _Key, _Tp, _Compare, _Alloc >, 2401
std::match_results< _Bi_iter, _Alloc >, 2414
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2495
std::multiset< _Key, _Compare, _Alloc >, 2521
std::set< _Key, _Compare, _Alloc >, 2949
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2076
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3191
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3222
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3248
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3276
std::vector< _Tp, _Alloc >, 3303
std::vector< bool, _Alloc >, 3323
max_size_type.h, 3425
mdspan, 3703
mean
std::normal_distribution< _RealType >, 2537
std::poisson_distribution< _IntType >, 2760
mem_fn
Function Objects, 308
Memory, 317
align, 318
assume_aligned, 318
uninitialized_copy, 319
uninitialized_copy_n, 319
uninitialized_default_construct, 320
uninitialized_default_construct_n, 320
uninitialized_fill, 320
uninitialized_fill_n, 321
uninitialized_move, 321
uninitialized_move_n, 322
uninitialized_value_construct, 322
uninitialized_value_construct_n, 322
memory, 3703–3705
memory_order
Atomics, 110
memory_resource, 3706, 3707
get_default_resource, 3707
null_memory_resource, 3707
set_default_resource, 3707
memory_resource.h, 3425
memoryfwd.h, 3425
merge
Sorting, 69, 70
std::forward_list< _Tp, _Alloc >, 2152
std::list< _Tp, _Alloc >, 2340
merge.h, 3733
merge_minimal_n
__gnu_parallel::_Settings, 1012
merge_oversampling
__gnu_parallel::_Settings, 1012
mersenne_twister_engine
std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >, 2427
message
std::error_category, 2111
std::error_code, 2113
std::error_condition, 2114
messages
std::locale, 2357
std::messages< _CharT >, 2431
messages_members.h, 3800
metadata_const_reference
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1724
__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1728
__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 935
__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 938
metadata_reference
__gnu_pbds::lu_counter_policy< Max_Count, _Alloc >, 2370
__gnu_pbds::lu_move_to_front_policy< _Alloc >, 2375
metadata_type
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1724
__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1729
__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 935
__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 938
__gnu_pbds::lu_counter_policy< Max_Count, _Alloc >, 2370
__gnu_pbds::lu_move_to_front_policy< _Alloc >, 2375
__gnu_pbds::sample_update_policy, 2857
Metaprogramming, 346
add_lvalue_reference_t, 350
add_pointer_t, 350
add_rvalue_reference_t, 350
aligned_storage_t, 350
alignment_value, 353
common_reference_t, 351

common_type_t, 351
conditional_t, 351
decay_t, 351
enable_if_t, 351
false_type, 351
is_corresponding_member, 352
is_layout_compatible_v, 353
is_pointer_interconvertible_base_of_v, 354
is_pointer_interconvertible_with_class, 353
make_signed_t, 351
make_unsigned_t, 351
remove_all_extents_t, 351
remove_extent_t, 352
remove_pointer_t, 352
remove_reference_t, 352
result_of_t, 352
swap, 353
true_type, 352
type, 352
underlying_type_t, 352
microseconds
Time, 365
milliseconds
Time, 365
min
__gnu_parallel, 452
Numeric Arrays, 223
Sorting, 70, 71
std::bernoulli_distribution, 1718
std::binomial_distribution< _IntType >, 1749
std::cauchy_distribution< _RealType >, 1775
std::chi_squared_distribution< _RealType >, 1798
std::discard_block_engine< _RandomNumberEngine, __p, __r >, 2059
std::discrete_distribution< _IntType >, 2062
std::exponential_distribution< _RealType >, 2120
std::extreme_value_distribution< _RealType >, 2124
std::fisher_f_distribution< _RealType >, 2133
std::gamma_distribution< _RealType >, 2180
std::geometric_distribution< _IntType >, 2183
std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >, 2244
std::linear_congruential_engine< _UIntType, __a, __c, __m >, 2320
std::lognormal_distribution< _RealType >, 2367
std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >, 2427
std::negative_binomial_distribution< _IntType >, 2531
std::normal_distribution< _RealType >, 2538
std::numeric_limits< _Tp >, 2575
std::numeric_limits< bool >, 2580
std::numeric_limits< char >, 2585
std::numeric_limits< char16_t >, 2591
std::numeric_limits< char32_t >, 2596
std::numeric_limits< double >, 2601
std::numeric_limits< float >, 2606
std::numeric_limits< int >, 2612
std::numeric_limits< long >, 2617
std::numeric_limits< long double >, 2622
std::numeric_limits< long long >, 2627
std::numeric_limits< short >, 2633
std::numeric_limits< signed char >, 2638
std::numeric_limits< unsigned char >, 2643
std::numeric_limits< unsigned int >, 2648
std::numeric_limits< unsigned long >, 2654
std::numeric_limits< unsigned long long >, 2659
std::numeric_limits< unsigned short >, 2664
std::numeric_limits< wchar_t >, 2669
std::piecewise_constant_distribution< _RealType >, 2744
std::piecewise_linear_distribution< _RealType >, 2748
std::poisson_distribution< _IntType >, 2760
std::shuffle_order_engine< _RandomNumberEngine, __k >, 2972
std::student_t_distribution< _RealType >, 3032
std::subtract_with_carry_engine< _UIntType, __w, __s, __r >, 3041
std::uniform_int_distribution< _IntType >, 3144
std::uniform_real_distribution< _RealType >, 3147
std::weibull_distribution< _RealType >, 3347
min_element
Sorting, 72
min_element_minimal_n
__gnu_parallel::_Settings, 1012
min_exponent
std::__numeric_limits_base, 790
std::numeric_limits< _Tp >, 2577
std::numeric_limits< bool >, 2582
std::numeric_limits< char >, 2588
std::numeric_limits< char16_t >, 2593
std::numeric_limits< char32_t >, 2598
std::numeric_limits< double >, 2603
std::numeric_limits< float >, 2609
std::numeric_limits< int >, 2614
std::numeric_limits< long >, 2619
std::numeric_limits< long double >, 2624
std::numeric_limits< long long >, 2630
std::numeric_limits< short >, 2635
std::numeric_limits< signed char >, 2640
std::numeric_limits< unsigned char >, 2645
std::numeric_limits< unsigned int >, 2651
std::numeric_limits< unsigned long >, 2656
std::numeric_limits< unsigned long long >, 2661
std::numeric_limits< unsigned short >, 2666
std::numeric_limits< wchar_t >, 2672

min_exponent10
std::__numeric_limits_base, 790
std::numeric_limits< _Tp >, 2577
std::numeric_limits< bool >, 2583
std::numeric_limits< char >, 2588
std::numeric_limits< char16_t >, 2593
std::numeric_limits< char32_t >, 2598
std::numeric_limits< double >, 2604
std::numeric_limits< float >, 2609
std::numeric_limits< int >, 2614
std::numeric_limits< long >, 2619
std::numeric_limits< long double >, 2625
std::numeric_limits< long long >, 2630
std::numeric_limits< short >, 2635
std::numeric_limits< signed char >, 2640
std::numeric_limits< unsigned char >, 2646
std::numeric_limits< unsigned int >, 2651
std::numeric_limits< unsigned long >, 2656
std::numeric_limits< unsigned long long >, 2661
std::numeric_limits< unsigned short >, 2667
std::numeric_limits< wchar_t >, 2672
minmax
Sorting, 72, 73
minmax_element
Sorting, 73
minstd_rand
Random Number Generators, 245
minstd_rand0
Random Number Generators, 245
minutes
Time, 365
mismatch
Non-Mutating, 55, 56
mod_based_range_hashing.hpp, 3637
modulus
std::linear_congruential_engine< _UIntType, __a, __c, __m >, 2323
mofunc_impl.h, 3425
monetary
std::locale, 2357
money_get
std::money_get< _CharT, _InIter >, 2442
money_put
std::money_put< _CharT, _OutIter >, 2447
moneypunct
std::moneypunct< _CharT, _Intl >, 2452
monostate.h, 3426
months
Time, 365
move
Mutating, 26
Utilities, 302
move.h, 3426
move_backward
Mutating, 26
move_if_noexcept
Utilities, 303
move_only_function
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>, 2470, 2471
mt19937
Random Number Generators, 245
mt19937_64
Random Number Generators, 245
mt_allocator.h, 3610
multiline
std::regex_constants, 718
multimap
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2479–2481
multimap.h, 3585
multiplier
std::linear_congruential_engine< _UIntType, __a, __c, __m >, 2323
multiseq_partition
__gnu_parallel, 453
multiseq_selection
__gnu_parallel, 453
multiseq_selection.h, 3734
multiset
std::multiset< _Key, _Compare, _Alloc >, 2507–2509
multiset.h, 3586
multiway_merge
__gnu_parallel, 454
multiway_merge.h, 3735
_GLIBCXX_PARALLEL_LENGTH, 3737
multiway_merge_3_variant
__gnu_parallel, 455
multiway_merge_4_variant
__gnu_parallel, 456
multiway_merge_exact_splitting
__gnu_parallel, 456
multiway_merge_loser_tree
__gnu_parallel, 457
multiway_merge_loser_tree_sentinel
__gnu_parallel, 457
multiway_merge_loser_tree_unguarded
__gnu_parallel, 458
multiway_merge_minimal_k
__gnu_parallel::_Settings, 1012
multiway_merge_minimal_n
__gnu_parallel::_Settings, 1012
multiway_merge_oversampling
__gnu_parallel::_Settings, 1012
multiway_merge_sampling_splitting
__gnu_parallel, 458

multiway_merge_sentinels
__gnu_parallel, 459
multiway_mergesort.h, 3737
Mutating, 19
copy, 21
copy_backward, 21
copy_if, 22
copy_n, 22
fill, 23
fill_n, 23
generate, 24
generate_n, 24
is_partitioned, 25
iter_swap, 25
move, 26
move_backward, 26
partition, 27
partition_copy, 27
partition_point, 29
random_shuffle, 29, 30
remove, 30
remove_copy, 31
remove_copy_if, 31
remove_if, 32
replace, 32
replace_copy_if, 33
replace_if, 33
reverse, 34
reverse_copy, 34
rotate, 35
rotate_copy, 35
shuffle, 36
stable_partition, 36
swap_ranges, 37
transform, 37, 38
unique, 38, 40
unique_copy, 40, 41
mutex, 3708
Mutexes, 116
adopt_lock, 118
call_once, 117
defer_lock, 118
lock, 117
try_lock, 117
try_to_lock, 118

name
__gnu_debug::type_info, 3133
std::error_category, 2111
std::locale, 2354
std::type_info, 3134
nanoseconds
Time, 365
narrow
std::__ctype_abstract_base< _CharT >, 757, 758
std::basic_fstream< _CharT, _Traits >, 1131
std::basic_ifstream< _CharT, _Traits >, 1181
std::basic_ios< _CharT, _Traits >, 1214
std::basic_iostream< _CharT, _Traits >, 1244
std::basic_istream< _CharT, _Traits >, 1289
std::basic_istream< _CharT, _Traits >::sentry, 2880
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1327
std::basic_ofstream< _CharT, _Traits >, 1363
std::basic_ostream< _CharT, _Traits >, 1394
std::basic_ostream< _CharT, _Traits >::sentry, 2908
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1427
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1687
std::ctype< _CharT >, 1918, 1919
std::ctype< char >, 1944–1946
std::ctype< wchar_t >, 1972, 1973
std::ctype_byname< _CharT >, 1986
std::ctype_byname< char >, 2005, 2006
native_handle
std::thread, 3053
neg_format
std::moneypunct< _CharT, _Intl >, 2456
std::moneypunct_byname< _CharT, _Intl >, 2464
negative_sign
std::moneypunct< _CharT, _Intl >, 2457
std::moneypunct_byname< _CharT, _Intl >, 2464
Negators, 316
not1, 316
not2, 317
nested_exception
std::nested_exception, 2533
nested_exception.h, 3427
nested_ptr
std::nested_exception, 2533
new, 3805
operator new, 3806
new_allocator.h, 3427, 3428
new_delete_resource
Polymorphic memory resources, 345
new_handler
std, 579
next_permutation
Sorting, 74
noboolalpha
std, 601
node.hpp, 3640, 3641
node_begin
__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >, 2697
__gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >, 2734

__gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >, 2804
__gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >, 2983
node_const_iterator
__gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >, 1731
__gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >, 1732
__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >, 3100
__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >, 3101
__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >, 3103
__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >, 3103
__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >, 3105
__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >, 3106
__gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >, 3121
__gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >, 3122
node_end
__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >, 2697
__gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >, 2735
__gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >, 2804
__gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >, 2983
node_handle.h, 3428
node_iterators.hpp, 3612
node_metadata_selector.hpp, 3650, 3651
node_type
__gnu_pbds::detail::pat_trie_base, 2732
__gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >, 2734
node_update
__gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >, 3121
__gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >, 3122
Non-Mutating, 42
adjacent_find, 44
all_of, 44
any_of, 45
count, 45
count_if, 46
equal, 46, 47
find, 48
find_end, 48, 49
find_first_of, 50
find_if, 51
find_if_not, 51
for_each, 52
for_each_n, 52
is_permutation, 53, 54
mismatch, 55, 56
none_of, 57
search, 57, 58
search_n, 58, 59
none
std::bitset< _Nb >, 1765
std::locale, 2357
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2076
none_of
Non-Mutating, 57
norm
Complex Numbers, 182
std::tr1, 726
Normal Distributions, 237
operator<<, 238
operator>>, 238
normal_distribution
std::normal_distribution< _RealType >, 2537
noshowbase
std, 602
noshowpoint
std, 602
noshowpos
std, 602
noskipws
std, 602
nosubs
std::regex_constants, 718
not1
Negators, 316
not2
Negators, 317
not_fn
std::experimental, 702
notify_cleared

__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1778
__gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >, 2225
__gnu_pbds::sample_resize_policy, 2849
__gnu_pbds::sample_resize_trigger, 2852
notify_erase_search_collision
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1778
__gnu_pbds::sample_resize_policy, 2849
__gnu_pbds::sample_resize_trigger, 2852
notify_erase_search_end
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1778
__gnu_pbds::sample_resize_policy, 2849
__gnu_pbds::sample_resize_trigger, 2852
notify_erase_search_start
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1778
__gnu_pbds::sample_resize_policy, 2849
__gnu_pbds::sample_resize_trigger, 2852
notify_erased
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1778
__gnu_pbds::sample_resize_policy, 2849
__gnu_pbds::sample_resize_trigger, 2852
notify_externally_resized
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1778
__gnu_pbds::sample_resize_trigger, 2852
notify_find_search_collision
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1778
__gnu_pbds::sample_resize_policy, 2849
__gnu_pbds::sample_resize_trigger, 2852
notify_find_search_end
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1779
__gnu_pbds::sample_resize_policy, 2850
__gnu_pbds::sample_resize_trigger, 2852
notify_find_search_start
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1779
__gnu_pbds::sample_resize_policy, 2850
__gnu_pbds::sample_resize_trigger, 2852
notify_insert_search_collision
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1779
__gnu_pbds::sample_resize_policy, 2850
__gnu_pbds::sample_resize_trigger, 2853
notify_insert_search_end
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1779
__gnu_pbds::sample_resize_policy, 2850
__gnu_pbds::sample_resize_trigger, 2853
notify_insert_search_start
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1779
__gnu_pbds::sample_resize_policy, 2850
__gnu_pbds::sample_resize_trigger, 2853
notify_inserted
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1779
__gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >, 2225
__gnu_pbds::sample_resize_policy, 2850
__gnu_pbds::sample_resize_trigger, 2853
notify_resized
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1779
__gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >, 2225
__gnu_pbds::sample_range_hashing, 2846
__gnu_pbds::sample_ranged_hash_fn, 2847
__gnu_pbds::sample_resize_policy, 2850
__gnu_pbds::sample_resize_trigger, 2853
nouppercase
std, 602
npos
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 858
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1545
std::basic_string< _CharT, _Traits, _Alloc >, 1644
nth_element
Sorting, 75
nth_element_minimal_n
__gnu_parallel::_Settings, 1013
null
memory_resource, 3707
null_node_metadata.hpp, 3629
nullopt
Optional values, 287
num
blocks
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2076
num_children
__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 936
__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 939
num_get
std::num_get< _CharT, _InIter >, 2547
num_put
std::num_put< _CharT, _OutIter >, 2563

numbers, 3708
numeric, 3708, 3709, 3711
    std::locale, 2357
Numeric Arrays, 216
    ~gslice, 221
    apply, 221, 222
    begin, 222
    cshift, 222
    end, 223
    gslice, 218
    gslice_array, 219
    indirect_array, 219
    mask_array, 219
    max, 223
    min, 223
    operator=, 223–227
    operator[], 227–230
    resize, 231
    shift, 231
    size, 232
    slice, 219
    slice_array, 220
    start, 232
    stride, 232
    sum, 232
    swap, 232
    valarray, 220, 221
numeric_traits.h, 3603
numericfwd.h, 3738
Numerics, 176
numpunct
    std::numpunct< _CharT >, 2674, 2675

oct
    std, 602
    std::basic_fstream< _CharT, _Traits >, 1158
    std::basic_ifstream< _CharT, _Traits >, 1201
    std::basic_ios< _CharT, _Traits >, 1223
    std::basic_iostream< _CharT, _Traits >, 1270
    std::basic_istream< _CharT, _Traits >, 1307
    std::basic_istream< _CharT, _Traits >::sentry, 2896
    std::basic_istringstream< _CharT, _Traits, _Alloc >, 1347
    std::basic_ofstream< _CharT, _Traits >, 1381
    std::basic_ostream< _CharT, _Traits >, 1411
    std::basic_ostream< _CharT, _Traits >::sentry, 2924
    std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1445
    std::basic_stringstream< _CharT, _Traits, _Alloc >, 1715
    std::ios_base, 2269
off_type
    std::basic_ios< _CharT, _Traits >, 1208
    std::basic_streambuf< _CharT, _Traits >, 1459
    std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3331
ofstream
    I/O, 165
omp_loop.h, 3739
omp_loop_static.h, 3740
once_flag
    std::once_flag, 2685
open
    __gnu_cxx::enc_filebuf< _CharT >, 2089, 2090
    __gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2995–2997
    std::basic_filebuf< _CharT, _Traits >, 1096, 1097
    std::basic_fstream< _CharT, _Traits >, 1131, 1132
    std::basic_ifstream< _CharT, _Traits >, 1182
    std::basic_ofstream< _CharT, _Traits >, 1363, 1364
openmode
    std::basic_fstream< _CharT, _Traits >, 1118
    std::basic_ifstream< _CharT, _Traits >, 1169
    std::basic_ios< _CharT, _Traits >, 1208
    std::basic_iostream< _CharT, _Traits >, 1231
    std::basic_istream< _CharT, _Traits >, 1277
    std::basic_istream< _CharT, _Traits >::sentry, 2869
    std::basic_istringstream< _CharT, _Traits, _Alloc >, 1316
    std::basic_ofstream< _CharT, _Traits >, 1354
    std::basic_ostream< _CharT, _Traits >, 1388
    std::basic_ostream< _CharT, _Traits >::sentry, 2903
    std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1419
    std::basic_stringstream< _CharT, _Traits, _Alloc >, 1676
    std::ios_base, 2260
operator _Iterator
    __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 967
    __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 981
operator _RAIter
    __gnu_parallel::_GuardedIterator< _RAIter, _Compare >, 887
operator bool
    std::basic_fstream< _CharT, _Traits >, 1132
    std::basic_ifstream< _CharT, _Traits >, 1183
    std::basic_ios< _CharT, _Traits >, 1214
    std::basic_iostream< _CharT, _Traits >, 1244
    std::basic_istream< _CharT, _Traits >, 1289
    std::basic_istream< _CharT, _Traits >::sentry, 2881
    std::basic_istringstream< _CharT, _Traits, _Alloc >, 1327
    std::basic_ofstream< _CharT, _Traits >, 1364
    std::basic_ostream< _CharT, _Traits >, 1395
    std::basic_ostream< _CharT, _Traits >::sentry, 2909
    std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1427

std::basic_stringstream< _CharT, _Traits, _Alloc >, 1688
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>, 1907
std::error_code, 2113
std::error_condition, 2114
std::function< _Res(_ArgTypes...)>, 2166
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>, 2471
std::shared_ptr< _Tp >, 2967
std::tr2::bool_set, 1771
std::unique_ptr< _Tp, _Dp >, 3153
std::unique_ptr< _Tp[], _Dp >, 3160
operator new
new, 3806
operator streamoff
std::fpos< _StateT >, 2158
operator string_type
std::sub_match< _BiIter >, 3037
operator!
std::basic_fstream< _CharT, _Traits >, 1132
std::basic_ifstream< _CharT, _Traits >, 1183
std::basic_ios< _CharT, _Traits >, 1215
std::basic_iostream< _CharT, _Traits >, 1244
std::basic_istream< _CharT, _Traits >, 1289
std::basic_istream< _CharT, _Traits >::sentry, 2881
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1327
std::basic_ofstream< _CharT, _Traits >, 1364
std::basic_ostream< _CharT, _Traits >, 1395
std::basic_ostream< _CharT, _Traits >::sentry, 2909
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1427
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1688
std::valarray< _Tp >, 3283
operator!=
__gnu_cxx, 395
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1725
__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1729
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1739
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1743
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >, 2306, 2307
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator< Node, _Alloc >, 2311
__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 936
__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 939
Dynamic Bitset., 135
std, 602, 603
std::rel_ops, 719
operator<
__gnu_cxx, 398
__gnu_parallel::_GuardedIterator< _RAIter, _Compare >, 887
Pointer Abstractions, 338, 339
std, 605–607
Time, 373
operator<<
Bernoulli Distributions, 235
Complex Numbers, 185
Diagnostics, 125
Dynamic Bitset., 136
Normal Distributions, 238
Pointer Abstractions, 339
Poisson Distributions, 239, 240
Random Number Generators, 246
Regular Expressions, 256
std, 608–617
std::__detail, 656
std::basic_fstream< _CharT, _Traits >, 1132–1134, 1136–1138
std::basic_iostream< _CharT, _Traits >, 1244–1249
std::basic_ofstream< _CharT, _Traits >, 1364–1366, 1368–1370
std::basic_ostream< _CharT, _Traits >, 1395–1400
std::basic_ostream< _CharT, _Traits >::sentry, 2909–2914
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1428–1432
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1688, 1689, 1691–1694
std::binomial_distribution< _IntType >, 1750
std::bitset< _Nb >, 1766
std::chi_squared_distribution< _RealType >, 1799
std::discard_block_engine< _RandomNumberEngine, __p, __r >, 2060
std::discrete_distribution< _IntType >, 2063
std::filesystem::path, 2742
std::fisher_f_distribution< _RealType >, 2133
std::gamma_distribution< _RealType >, 2181
std::linear_congruential_engine< _UIntType, __a, __c, __m >, 2321
std::lognormal_distribution< _RealType >, 2368
std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >, 2428
std::negative_binomial_distribution< _IntType >, 2532
std::normal_distribution< _RealType >, 2539

std::piecewise_constant_distribution< _RealType >, 2746
std::piecewise_linear_distribution< _RealType >, 2749
std::poisson_distribution< _IntType >, 2761
std::shuffle_order_engine< _RandomNumberEngine, __k >, 2972
std::student_t_distribution< _RealType >, 3033
std::subtract_with_carry_engine< _UIntType, __w, __s, __r >, 3042
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2077
std::unique_ptr< _Tp[], _Dp >, 3163
Time, 373
Uniform Distributions, 243
operator<<=
std::bitset< _Nb >, 1766
std::gslice_array< _Tp >, 2205
std::indirect_array< _Tp >, 2247
std::mask_array< _Tp >, 2407
std::slice_array< _Tp >, 2976
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2077
std::valarray< _Tp >, 3285
operator<=
__gnu_cxx, 399, 400
__gnu_parallel::_GuardedIterator< _RAIter, _Compare >, 887
Dynamic Bitset., 136
Pointer Abstractions, 339, 340
std, 617, 618
std::rel_ops, 719
Time, 374
operator<=>
__gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >, 896, 897
Diagnostics, 125
Regular Expressions, 256, 257
std, 618–620
std::error_category, 2111
std::filesystem::path, 2742
Time, 374
Utilities, 304
operator>
__gnu_cxx, 401, 402
Dynamic Bitset., 136
Pointer Abstractions, 340, 341
std, 626, 627
std::rel_ops, 719
Time, 374
operator>>
Bernoulli Distributions, 235, 237
Complex Numbers, 185
Dynamic Bitset., 137
Normal Distributions, 238
Poisson Distributions, 240, 241
std, 627–632
std::__detail, 656
std::basic_fstream< _CharT, _Traits >, 1138–1143
std::basic_ifstream< _CharT, _Traits >, 1183–1188
std::basic_iostream< _CharT, _Traits >, 1250–1255
std::basic_istream< _CharT, _Traits >, 1289–1294
std::basic_istream< _CharT, _Traits >::sentry, 2881–2886
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1328, 1329, 1331–1333
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1694, 1695, 1697–1700
std::binomial_distribution< _IntType >, 1752
std::bitset< _Nb >, 1766
std::chi_squared_distribution< _RealType >, 1799
std::discard_block_engine< _RandomNumberEngine, __p, __r >, 2060
std::discrete_distribution< _IntType >, 2064
std::filesystem::path, 2742
std::fisher_f_distribution< _RealType >, 2134
std::gamma_distribution< _RealType >, 2182
std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >, 2245
std::linear_congruential_engine< _UIntType, __a, __c, __m >, 2321
std::lognormal_distribution< _RealType >, 2368
std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >, 2428
std::negative_binomial_distribution< _IntType >, 2532
std::normal_distribution< _RealType >, 2539
std::piecewise_constant_distribution< _RealType >, 2746
std::piecewise_linear_distribution< _RealType >, 2750
std::poisson_distribution< _IntType >, 2762
std::shuffle_order_engine< _RandomNumberEngine, __k >, 2973
std::student_t_distribution< _RealType >, 3034
std::subtract_with_carry_engine< _UIntType, __w, __s, __r >, 3042
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2078
Uniform Distributions, 243, 244
operator>>=
std::bitset< _Nb >, 1766
std::gslice_array< _Tp >, 2205
std::indirect_array< _Tp >, 2247
std::mask_array< _Tp >, 2407
std::slice_array< _Tp >, 2976
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2078
std::valarray< _Tp >, 3285
operator>=
__gnu_cxx, 402, 403

Dynamic Bitset., 136
Pointer Abstractions, 341
std, 627
std::rel_ops, 720
Time, 374
operator()
__gnu_cxx::subtractive_rng, 3044
__gnu_parallel::_Nothing, 941
__gnu_parallel::_RandomNumber, 952
__gnu_parallel::__accumulate_selector< _It >, 730
__gnu_parallel::__adjacent_find_selector, 732
__gnu_parallel::__count_if_selector< _It, _Diff >, 747
__gnu_parallel::__count_selector< _It, _Diff >, 748
__gnu_parallel::__fill_selector< _It >, 765
__gnu_parallel::__find_first_of_selector< _FIterator >, 766
__gnu_parallel::__find_if_selector, 768
__gnu_parallel::__for_each_selector< _It >, 769
__gnu_parallel::__generate_selector< _It >, 770
__gnu_parallel::__identity_selector< _It >, 774
__gnu_parallel::__inner_product_selector< _It, _It2, _Tp >, 775
__gnu_parallel::__mismatch_selector, 779
__gnu_parallel::__replace_if_selector< _It, _Op, _Tp >, 801
__gnu_parallel::__replace_selector< _It, _Tp >, 803
__gnu_parallel::__transform1_selector< _It >, 804
__gnu_parallel::__transform2_selector< _It >, 805
__gnu_pbds::direct_mask_range_hashing< Size_Type >, 2052
__gnu_pbds::direct_mod_range_hashing< Size_Type >, 2053
__gnu_pbds::linear_probe_fn< Size_Type >, 2324
__gnu_pbds::lu_counter_policy< Max_Count, _Alloc >, 2371
__gnu_pbds::lu_move_to_front_policy< _Alloc >, 2375
__gnu_pbds::quadratic_probe_fn< Size_Type >, 2775
__gnu_pbds::sample_probe_fn, 2845
__gnu_pbds::sample_range_hashing, 2846
__gnu_pbds::sample_ranged_hash_fn, 2847
__gnu_pbds::sample_trie_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >, 2856
__gnu_pbds::sample_update_policy, 2857
__gnu_pbds::tree_order_statistics_node_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc >, 3098
__gnu_pbds::trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >, 3112
__gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >, 3116
std::bernoulli_distribution, 1718
std::binomial_distribution< _IntType >, 1749
std::cauchy_distribution< _RealType >, 1775
std::chi_squared_distribution< _RealType >, 1798
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>, 1907
std::default_delete< _Tp >, 2020
std::default_delete< _Tp[]>, 2021
std::discard_block_engine< _RandomNumberEngine, __p, __r >, 2059
std::discrete_distribution< _IntType >, 2062
std::exponential_distribution< _RealType >, 2120
std::extreme_value_distribution< _RealType >, 2124
std::fisher_f_distribution< _RealType >, 2133
std::function< _Res(_ArgTypes...)>, 2166
std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>, 2172
std::gamma_distribution< _RealType >, 2180, 2181
std::geometric_distribution< _IntType >, 2184
std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >, 2244
std::linear_congruential_engine< _UIntType, __a, __c, __m >, 2320
std::locale, 2354
std::lognormal_distribution< _RealType >, 2367
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>, 2471
std::negative_binomial_distribution< _IntType >, 2531
std::normal_distribution< _RealType >, 2538
std::piecewise_constant_distribution< _RealType >, 2744
std::piecewise_linear_distribution< _RealType >, 2749
std::plus< _Tp >, 2752
std::poisson_distribution< _IntType >, 2760, 2761
std::shuffle_order_engine< _RandomNumberEngine, __k >, 2972
std::student_t_distribution< _RealType >, 3032
std::subtract_with_carry_engine< _UIntType, __w, __s, __r >, 3041
std::uniform_int_distribution< _IntType >, 3144
std::uniform_real_distribution< _RealType >, 3147
std::weibull_distribution< _RealType >, 3347
operator+
__gnu_cxx, 396, 397
Complex Numbers, 183
std, 604, 605
std::fpos< _StateT >, 2158
std::reverse_iterator< _Iterator >, 2836
std::valarray< _Tp >, 3284
Time, 372
operator++
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 967, 968
__gnu_debug::_Safe_local_iterator< _Iterator,

_UContainer >, 982
__gnu_parallel::_GuardedIterator< _RAIter, _Compare >, 887
std::back_insert_iterator< _Container >, 1078
std::front_insert_iterator< _Container >, 2163
std::insert_iterator< _Container >, 2253
std::istreambuf_iterator< _CharT, _Traits >, 2296
std::ostreambuf_iterator< _CharT, _Traits >, 2693
std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >, 2818, 2819
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >, 2823
std::reverse_iterator< _Iterator >, 2836
operator+=
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 841, 843
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1526
Complex Numbers, 183
std::basic_string< _CharT, _Traits, _Alloc >, 1615–1617
std::complex< _Tp >, 1867
std::complex< double >, 1869
std::complex< float >, 1872
std::complex< long double >, 1875
std::fpos< _StateT >, 2158
std::gslice_array< _Tp >, 2205
std::indirect_array< _Tp >, 2247
std::mask_array< _Tp >, 2407
std::reverse_iterator< _Iterator >, 2837
std::slice_array< _Tp >, 2976
std::valarray< _Tp >, 3284
operator-
Complex Numbers, 183, 184
Dynamic Bitset., 135
std::fpos< _StateT >, 2158, 2159
std::reverse_iterator< _Iterator >, 2837
std::valarray< _Tp >, 3284
Time, 372
operator->
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 968
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 982
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1740
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1743
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >, 2307
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >, 2311
std::auto_ptr< _Tp >, 1074
std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >, 2819
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >, 2823
std::reverse_iterator< _Iterator >, 2838
std::unique_ptr< _Tp, _Dp >, 3153
std::unique_ptr< _Tp[], _Dp >, 3161
operator--
std::reverse_iterator< _Iterator >, 2837
operator-=
Complex Numbers, 184
std::complex< _Tp >, 1867
std::complex< double >, 1869
std::complex< float >, 1872
std::complex< long double >, 1875
std::fpos< _StateT >, 2159
std::gslice_array< _Tp >, 2205
std::indirect_array< _Tp >, 2247
std::mask_array< _Tp >, 2407
std::reverse_iterator< _Iterator >, 2837
std::slice_array< _Tp >, 2976
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2077
std::valarray< _Tp >, 3284
operator/
Complex Numbers, 184
std::filesystem::path, 2742
Time, 373
operator/=
Complex Numbers, 184, 185
std::complex< double >, 1869, 1870
std::complex< float >, 1872
std::complex< long double >, 1875
std::gslice_array< _Tp >, 2205
std::indirect_array< _Tp >, 2247
std::mask_array< _Tp >, 2407
std::slice_array< _Tp >, 2976
std::valarray< _Tp >, 3285
operator=
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 844, 845
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 968
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 982
Complex Numbers, 185
Numeric Arrays, 223–227
std::auto_ptr< _Tp >, 1074, 1075
std::back_insert_iterator< _Container >, 1078
std::basic_regex< _Ch_type, _Rx_traits >, 1454
std::basic_string< _CharT, _Traits, _Alloc >, 1617–1620
std::complex< double >, 1870
std::complex< float >, 1872, 1873
std::complex< long double >, 1875

std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>, 1907, 1908
std::deque< _Tp, _Alloc >, 2046, 2047
std::experimental::fundamentals_v1::any, 1065, 1066
std::forward_list< _Tp, _Alloc >, 2152, 2153
std::front_insert_iterator< _Container >, 2163
std::function< _Res(_ArgTypes...)>, 2167, 2168
std::insert_iterator< _Container >, 2254
std::list< _Tp, _Alloc >, 2341
std::locale, 2355
std::map< _Key, _Tp, _Compare, _Alloc >, 2401, 2402
std::match_results< _Bi_iter, _Alloc >, 2415
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>, 2471, 2472
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2496
std::multiset< _Key, _Compare, _Alloc >, 2521
std::once_flag, 2685
std::ostream_iterator< _Tp, _CharT, _Traits >, 2689
std::ostreambuf_iterator< _CharT, _Traits >, 2693
std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >, 2819
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >, 2823
std::set< _Key, _Compare, _Alloc >, 2949
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2078
std::unique_ptr< _Tp, _Dp >, 3153
std::unique_ptr< _Tp[], _Dp >, 3161, 3162
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3191, 3192
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3222
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3248, 3249
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3276, 3277
std::vector< _Tp, _Alloc >, 3303, 3304
std::vector< bool, _Alloc >, 3324
operator==
__gnu_cxx, 400, 401
__gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >, 897, 898
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1725
__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1730
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1740
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1743
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >, 2307
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_ite
Node, _Alloc >, 2311
__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 936
__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 939
Allocators, 324
Complex Numbers, 185
Diagnostics, 125, 126
Iterators, 172
Pointer Abstractions, 340
Regular Expressions, 257, 258
std, 621–626
std::__fwdlist::_Iterator< _Const, _Ptr >, 893
std::_Fwd_list_const_iterator< _Tp >, 883
std::_Fwd_list_iterator< _Tp >, 884
std::bernoulli_distribution, 1719
std::binomial_distribution< _IntType >, 1752
std::bitset< _Nb >, 1766
std::cauchy_distribution< _RealType >, 1776
std::chi_squared_distribution< _RealType >, 1799
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>, 1908
std::discard_block_engine< _RandomNumberEngine, __p, __r >, 2060
std::discrete_distribution< _IntType >, 2063
std::error_category, 2111
std::exponential_distribution< _RealType >, 2121
std::extreme_value_distribution< _RealType >, 2125
std::filesystem::path, 2742
std::fisher_f_distribution< _RealType >, 2134
std::gamma_distribution< _RealType >, 2182
std::geometric_distribution< _IntType >, 2184
std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >, 2244
std::istream_iterator< _Tp, _CharT, _Traits, _Dist >, 2292
std::linear_congruential_engine< _UIntType, __a, __c, __m >, 2321
std::locale, 2355
std::lognormal_distribution< _RealType >, 2368
std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >, 2428
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>, 2472
std::negative_binomial_distribution< _IntType >, 2532
std::normal_distribution< _RealType >, 2539
std::piecewise_constant_distribution< _RealType >, 2746
std::piecewise_linear_distribution< _RealType >,

2750
std::poisson_distribution< _IntType >, 2762
std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >, 2819
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >, 2823
std::shuffle_order_engine< _RandomNumberEngine, __k >, 2973
std::slice, 2974
std::student_t_distribution< _RealType >, 3033
std::subtract_with_carry_engine< _UIntType, __w, __s, __r >, 3042
std::uniform_int_distribution< _IntType >, 3145
std::uniform_real_distribution< _RealType >, 3148
std::weibull_distribution< _RealType >, 3348
Time, 374
Utilities, 304
operator%
Time, 371
operator%=
std::gslice_array< _Tp >, 2204
std::indirect_array< _Tp >, 2247
std::mask_array< _Tp >, 2406
std::slice_array< _Tp >, 2976
std::valarray< _Tp >, 3283
operator&
Dynamic Bitset., 135
std, 603
std::regex_constants, 712
operator&=
std::bitset< _Nb >, 1766
std::gslice_array< _Tp >, 2204
std::indirect_array< _Tp >, 2247
std::mask_array< _Tp >, 2406
std::regex_constants, 712, 713
std::slice_array< _Tp >, 2976
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2077
std::valarray< _Tp >, 3283
operator[]
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 845
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1526, 1527
Numeric Arrays, 227–230
std::basic_string< _CharT, _Traits, _Alloc >, 1620, 1621
std::bitset< _Nb >, 1767
std::deque< _Tp, _Alloc >, 2047, 2048
std::map< _Key, _Tp, _Compare, _Alloc >, 2402
std::match_results< _Bi_iter, _Alloc >, 2415
std::reverse_iterator< _Iterator >, 2838
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2078, 2079
std::unique_ptr< _Tp[], _Dp >, 3162
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3192
std::vector< _Tp, _Alloc >, 3304
std::vector< bool, _Alloc >, 3324, 3325
operator*
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 967
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 981
__gnu_parallel::_GuardedIterator< _RAIter, _Compare >, 887
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1725
__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1730
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1740
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1743
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >, 2307
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_ite
Node, _Alloc >, 2311
__gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >, 2701
__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 936
__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 939
Complex Numbers, 182
std::auto_ptr< _Tp >, 1074
std::back_insert_iterator< _Container >, 1078
std::front_insert_iterator< _Container >, 2163
std::insert_iterator< _Container >, 2253
std::istreambuf_iterator< _CharT, _Traits >, 2296
std::ostreambuf_iterator< _CharT, _Traits >, 2693
std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >, 2818
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >, 2823
std::reverse_iterator< _Iterator >, 2836
std::unique_ptr< _Tp, _Dp >, 3153
std::unique_ptr< _Tp[], _Dp >, 3161
Time, 371
operator*=
Complex Numbers, 182
std::complex< double >, 1869
std::complex< float >, 1872
std::complex< long double >, 1874
std::gslice_array< _Tp >, 2205
std::indirect_array< _Tp >, 2247
std::mask_array< _Tp >, 2407

std::slice_array< _Tp >, 2976
std::valarray< _Tp >, 3284
operator~
std::bitset< _Nb >, 1768
std::regex_constants, 714
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2080
std::valarray< _Tp >, 3286
operator^
Dynamic Bitset., 137
std, 632
std::regex_constants, 713
operator^=
std::bitset< _Nb >, 1767
std::gslice_array< _Tp >, 2205
std::indirect_array< _Tp >, 2248
std::mask_array< _Tp >, 2407
std::regex_constants, 713
std::slice_array< _Tp >, 2977
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2079
std::valarray< _Tp >, 3285
operator|
Dynamic Bitset., 137
std, 633
std::regex_constants, 714
operator|=
std::bitset< _Nb >, 1768
std::gslice_array< _Tp >, 2205
std::indirect_array< _Tp >, 2248
std::mask_array< _Tp >, 2407
std::regex_constants, 714
std::slice_array< _Tp >, 2977
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2079
std::valarray< _Tp >, 3286
opt_random.h, 3800
optimize
std::regex_constants, 718
optional, 3713
Optional values, 286
in_place, 287
nullopt, 287
order_of_key
__gnu_pbds::tree_order_statistics_node_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc >, 3098
__gnu_pbds::trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >, 3112
order_of_prefix
__gnu_pbds::trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >, 3112
order_preserving
__gnu_pbds::container_traits< Cntnr >, 1899
order_statistics_imp.hpp, 3651
os_defines.h, 3800
ostream, 3713
I/O, 165
ostream.h, 3428
ostream.tcc, 3430
ostream_insert.h, 3430
ostream_iterator
std::ostream_iterator< _Tp, _CharT, _Traits >, 2689
ostream_type
std::ostream_iterator< _Tp, _CharT, _Traits >, 2688
std::ostreambuf_iterator< _CharT, _Traits >, 2691
ostreambuf_iterator
std::ostreambuf_iterator< _CharT, _Traits >, 2692
ostringstream
I/O, 165
out
std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >, 744
std::basic_fstream< _CharT, _Traits >, 1158
std::basic_ifstream< _CharT, _Traits >, 1201
std::basic_ios< _CharT, _Traits >, 1223
std::basic_iostream< _CharT, _Traits >, 1270
std::basic_istream< _CharT, _Traits >, 1307
std::basic_istream< _CharT, _Traits >::sentry, 2896
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1347
std::basic_ofstream< _CharT, _Traits >, 1381
std::basic_ostream< _CharT, _Traits >, 1411
std::basic_ostream< _CharT, _Traits >::sentry, 2924
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1445
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1715
std::codecvt< _InternT, _ExternT, _StateT >, 1803
std::codecvt< _InternT, _ExternT, encoding_state >, 1811, 1812
std::codecvt< char, char, mbstate_t >, 1820
std::codecvt< char16_t, char, mbstate_t >, 1828, 1829
std::codecvt< char32_t, char, mbstate_t >, 1837, 1838
std::codecvt< wchar_t, char, mbstate_t >, 1847
std::codecvt_byname< _InternT, _ExternT, _StateT >, 1854
std::ios_base, 2269
out_ptr.h, 3430
ov_tree_map_.hpp, 3642
overflow
__gnu_cxx::enc_filebuf< _CharT >, 2090
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2997
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3015
std::basic_filebuf< _CharT, _Traits >, 1098
std::basic_streambuf< _CharT, _Traits >, 1462
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1657
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3334
owner_before

std::shared_ptr< _Tp >, 2967, 2968

p
std::bernoulli_distribution, 1718
std::binomial_distribution< _IntType >, 1750
std::geometric_distribution< _IntType >, 2184
std::negative_binomial_distribution< _IntType >, 2531
pair
__gnu_cxx::pair< _T1, _T2 >, 2710, 2711
std::pair< _T1, _T2 >, 2714
Utilities, 298
pairing_heap_.hpp, 3643
par_loop.h, 3740
parallel.h, 3740
parallel_balanced
__gnu_parallel, 423
parallel_multiway_merge
__gnu_parallel, 460
parallel_omp_loop
__gnu_parallel, 423
parallel_omp_loop_static
__gnu_parallel, 423
parallel_sort_mwms
__gnu_parallel, 461
parallel_sort_mwms_pu
__gnu_parallel, 461
parallel_tag
__gnu_parallel::parallel_tag, 2720
parallel_taskqueue
__gnu_parallel, 423
parallel_unbalanced
__gnu_parallel, 423
Parallelism TS, 294
param
std::bernoulli_distribution, 1718
std::binomial_distribution< _IntType >, 1750
std::cauchy_distribution< _RealType >, 1775
std::chi_squared_distribution< _RealType >, 1798
std::discrete_distribution< _IntType >, 2062, 2063
std::exponential_distribution< _RealType >, 2120, 2121
std::extreme_value_distribution< _RealType >, 2124
std::fisher_f_distribution< _RealType >, 2133
std::gamma_distribution< _RealType >, 2181
std::geometric_distribution< _IntType >, 2184
std::lognormal_distribution< _RealType >, 2367
std::negative_binomial_distribution< _IntType >, 2531
std::normal_distribution< _RealType >, 2538
std::piecewise_constant_distribution< _RealType >, 2744
std::piecewise_linear_distribution< _RealType >, 2749
std::poisson_distribution< _IntType >, 2761
std::student_t_distribution< _RealType >, 3033
std::uniform_int_distribution< _IntType >, 3145
std::uniform_real_distribution< _RealType >, 3147
std::weibull_distribution< _RealType >, 3348
parse_numbers.h, 3430
partial_sort
Sorting, 76
partial_sort_copy
Sorting, 77
partial_sort_minimal_n
__gnu_parallel::_Settings, 1013
partial_sum
Generalized Numeric operations, 12, 13
partial_sum.h, 3741
partial_sum_dilation
__gnu_parallel::_Settings, 1013
partial_sum_minimal_n
__gnu_parallel::_Settings, 1013
partition
Mutating, 27
partition.h, 3741
_GLIBCXX_VOLATILE, 3741
partition_chunk_share
__gnu_parallel::_Settings, 1013
partition_chunk_size
__gnu_parallel::_Settings, 1013
partition_copy
Mutating, 27
partition_minimal_n
__gnu_parallel::_Settings, 1013
partition_point
Mutating, 29
pat_trie_.hpp, 3643
pat_trie_base.hpp, 3644
pbackfail
__gnu_cxx::enc_filebuf< _CharT >, 2091
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2998
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3015
std::basic_filebuf< _CharT, _Traits >, 1098
std::basic_streambuf< _CharT, _Traits >, 1463
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1657
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3335
pbase
__gnu_cxx::enc_filebuf< _CharT >, 2091
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2998
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3016
std::basic_filebuf< _CharT, _Traits >, 1099
std::basic_streambuf< _CharT, _Traits >, 1463
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1658
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3335
pbump

__gnu_cxx::enc_filebuf< _CharT >, 2092
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2998
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3016
std::basic_filebuf< _CharT, _Traits >, 1099
std::basic_streambuf< _CharT, _Traits >, 1464
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1658
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3335
peek
std::basic_fstream< _CharT, _Traits >, 1144
std::basic_ifstream< _CharT, _Traits >, 1188
std::basic_iostream< _CharT, _Traits >, 1255
std::basic_istream< _CharT, _Traits >, 1295
std::basic_istream< _CharT, _Traits >::sentry, 2886
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1334
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1701
perm_options
Filesystem, 162
perms
Filesystem, 162
Filesystem TS, 279
piecewise_construct
Utilities, 306
pod_char_traits.h, 3658
point_const_iterator.hpp, 3625
point_iterator.hpp, 3654
point_iterators.hpp, 3612
pointer
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1739
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1742
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >, 2306
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >, 2310
std::allocator_traits< _Alloc >, 1033
std::allocator_traits< allocator< _Tp > >, 1039
std::allocator_traits< allocator< void > >, 1048
std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1056
std::back_insert_iterator< _Container >, 1077
std::front_insert_iterator< _Container >, 2162
std::insert_iterator< _Container >, 2253
std::istream_iterator< _Tp, _CharT, _Traits, _Dist >, 2292
std::istreambuf_iterator< _CharT, _Traits >, 2295
std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >, 2300
std::ostream_iterator< _Tp, _CharT, _Traits >, 2688
std::ostreambuf_iterator< _CharT, _Traits >, 2692
std::pointer_traits< _Ptr >, 2755
std::pointer_traits< _Tp * >, 2757
std::raw_storage_iterator< _OutputIterator, _Tp >, 2797
std::set< _Key, _Compare, _Alloc >, 2933
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3172
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3203
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3231
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3258
Pointer Abstractions, 325
allocate_shared, 328
atomic_compare_exchange_strong, 329
atomic_compare_exchange_strong_explicit, 330
atomic_compare_exchange_weak, 331
atomic_compare_exchange_weak_explicit, 332
atomic_exchange, 333
atomic_exchange_explicit, 333, 334
atomic_is_lock_free, 334
atomic_load, 335
atomic_load_explicit, 335, 336
atomic_store, 336
atomic_store_explicit, 337
const_pointer_cast, 337
dynamic_pointer_cast, 338
get_deleter, 338
make_shared, 338
operator<, 338, 339
operator<<, 339
operator<=, 339, 340
operator>, 340, 341
operator>=, 341
operator==, 340
reinterpret_pointer_cast, 341
static_pointer_cast, 342
swap, 342
to_address, 342, 343
Pointer Safety and Garbage Collection, 343
declare_no_pointers, 344
declare_reachable, 344
get_pointer_safety, 344
pointer_safety, 344
undeclare_no_pointers, 344
undeclare_reachable, 344
pointer.h, 3658
pointer_safety
Pointer Safety and Garbage Collection, 344
pointer_to
std::pointer_traits< _Ptr >, 2756
std::pointer_traits< _Tp * >, 2758
Poisson Distributions, 239
operator<<, 239, 240

operator>>, 240, 241
polar
Complex Numbers, 186
std::tr1, 726
Policy-Based Data Structures, 138
policy_access_fn_imps.hpp, 3626
Polymorphic memory resources, 345
new_delete_resource, 345
pool_allocator.h, 3660
pop
std::priority_queue< _Tp, _Sequence, _Compare >, 2768
std::queue< _Tp, _Sequence >, 2778
std::stack< _Tp, _Sequence >, 2987
pop_back
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 846
__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >, 955
std::basic_string< _CharT, _Traits, _Alloc >, 1621
std::deque< _Tp, _Alloc >, 2048
std::list< _Tp, _Alloc >, 2341
std::vector< _Tp, _Alloc >, 3305
std::vector< bool, _Alloc >, 3325
pop_front
__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >, 955
std::deque< _Tp, _Alloc >, 2048
std::forward_list< _Tp, _Alloc >, 2153
std::list< _Tp, _Alloc >, 2342
pop_heap
Heap, 89, 90
pos_format
std::moneypunct< _CharT, _Intl >, 2457
std::moneypunct_byname< _CharT, _Intl >, 2465
pos_type
std::basic_ios< _CharT, _Traits >, 1208
std::basic_streambuf< _CharT, _Traits >, 1459
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3331
position
std::match_results< _Bi_iter, _Alloc >, 2415
positive_sign
std::moneypunct< _CharT, _Intl >, 2457
std::moneypunct_byname< _CharT, _Intl >, 2465
postypes.h, 3431
pow
Complex Numbers, 186
math.h, 3702
power
SGI, 155
pptr
__gnu_cxx::enc_filebuf< _CharT >, 2092
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2999
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3016
std::basic_filebuf< _CharT, _Traits >, 1099
std::basic_streambuf< _CharT, _Traits >, 1464
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1658
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3336
precision
std::basic_fstream< _CharT, _Traits >, 1144
std::basic_ifstream< _CharT, _Traits >, 1188, 1189
std::basic_ios< _CharT, _Traits >, 1215
std::basic_iostream< _CharT, _Traits >, 1255, 1256
std::basic_istream< _CharT, _Traits >, 1295
std::basic_istream< _CharT, _Traits >::sentry, 2886
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1334
std::basic_ofstream< _CharT, _Traits >, 1370, 1371
std::basic_ostream< _CharT, _Traits >, 1400
std::basic_ostream< _CharT, _Traits >::sentry, 2914
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1433
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1701
std::ios_base, 2262, 2263
predefined_ops.h, 3431
prefix
std::match_results< _Bi_iter, _Alloc >, 2416
prefix_range
__gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >, 3116, 3117
prefix_search_node_update_imp.hpp, 3651
prev_permutation
Sorting, 78
print, 3748
priority_queue
__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >, 2765
std::priority_queue< _Tp, _Sequence, _Compare >, 2767
priority_queue.hpp, 3655
priority_queue_base_dispatch.hpp, 3646
probabilities
std::discrete_distribution< _IntType >, 2063
probe_fn_base.hpp, 3638
propagate_const, 3604
propagate_on_container_copy_assignment
__gnu_cxx::__alloc_traits< _Alloc, typename >, 734
std::allocator_traits< _Alloc >, 1033
std::allocator_traits< allocator< _Tp > >, 1039
std::allocator_traits< allocator< void > >, 1048
std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1056, 1057
propagate_on_container_move_assignment
__gnu_cxx::__alloc_traits< _Alloc, typename >, 734

std::allocator_traits< _Alloc >, 1033
std::allocator_traits< allocator< _Tp > >, 1039
std::allocator_traits< allocator< void > >, 1048
std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1057
propagate_on_container_swap
__gnu_cxx::__alloc_traits< _Alloc, typename >, 735
std::allocator_traits< _Alloc >, 1033
std::allocator_traits< allocator< _Tp > >, 1040
std::allocator_traits< allocator< void > >, 1048
std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1057
ptr_fun
Adaptors for pointers to functions, 309
ptr_traits.h, 3432
pubimbue
__gnu_cxx::enc_filebuf< _CharT >, 2092
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2999
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3016
std::basic_filebuf< _CharT, _Traits >, 1100
std::basic_streambuf< _CharT, _Traits >, 1464
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1659
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3336
pubseekoff
__gnu_cxx::enc_filebuf< _CharT >, 2092
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2999
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3017
std::basic_filebuf< _CharT, _Traits >, 1100
std::basic_streambuf< _CharT, _Traits >, 1465
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1659
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3336
pubseekpos
__gnu_cxx::enc_filebuf< _CharT >, 2093
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 3000
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3017
std::basic_filebuf< _CharT, _Traits >, 1100
std::basic_streambuf< _CharT, _Traits >, 1465
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1659
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3337
pubsetbuf
__gnu_cxx::enc_filebuf< _CharT >, 2093
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 3000
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3017
std::basic_filebuf< _CharT, _Traits >, 1101
std::basic_streambuf< _CharT, _Traits >, 1465
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1660
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3337
pubsync
__gnu_cxx::enc_filebuf< _CharT >, 2093
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 3000
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3018
std::basic_filebuf< _CharT, _Traits >, 1101
std::basic_streambuf< _CharT, _Traits >, 1465
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1660
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3337
push
std::priority_queue< _Tp, _Sequence, _Compare >, 2768
std::queue< _Tp, _Sequence >, 2778
std::stack< _Tp, _Sequence >, 2987
push_back
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 846
std::basic_string< _CharT, _Traits, _Alloc >, 1621, 1622
std::deque< _Tp, _Alloc >, 2048
std::list< _Tp, _Alloc >, 2342
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2080
std::vector< _Tp, _Alloc >, 3305
std::vector< bool, _Alloc >, 3325
push_front
__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >, 956
std::deque< _Tp, _Alloc >, 2049
std::forward_list< _Tp, _Alloc >, 2153
std::list< _Tp, _Alloc >, 2342
push_heap
Heap, 90, 91
put
std::basic_fstream< _CharT, _Traits >, 1144
std::basic_iostream< _CharT, _Traits >, 1256
std::basic_ofstream< _CharT, _Traits >, 1371
std::basic_ostream< _CharT, _Traits >, 1401
std::basic_ostream< _CharT, _Traits >::sentry, 2916
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1433
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1701
std::money_put< _CharT, _OutIter >, 2448, 2449
std::num_put< _CharT, _OutIter >, 2567–2572
std::time_put< _CharT, _OutIter >, 3087, 3088
std::time_put_byname< _CharT, _OutIter >, 3090, 3091
put_money
std, 633
put_time
std, 633
putback
std::basic_fstream< _CharT, _Traits >, 1145
std::basic_ifstream< _CharT, _Traits >, 1189
std::basic_iostream< _CharT, _Traits >, 1256
std::basic_istream< _CharT, _Traits >, 1295
std::basic_istream< _CharT, _Traits >::sentry, 2886

std::basic_istringstream< _CharT, _Traits, _Alloc >, 1334
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1702
pword
std::basic_fstream< _CharT, _Traits >, 1145
std::basic_ifstream< _CharT, _Traits >, 1189
std::basic_ios< _CharT, _Traits >, 1215
std::basic_iostream< _CharT, _Traits >, 1257
std::basic_istream< _CharT, _Traits >, 1296
std::basic_istream< _CharT, _Traits >::sentry, 2887
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1335
std::basic_ofstream< _CharT, _Traits >, 1371
std::basic_ostream< _CharT, _Traits >, 1401
std::basic_ostream< _CharT, _Traits >::sentry, 2916
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1434
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1702
std::ios_base, 2263

qsb_steals
__gnu_parallel::_Settings, 1013
quadratic_probe_fn_imp.hpp, 3638
queue, 3748
std::queue< _Tp, _Sequence >, 2777
queue.h, 3742
_GLIBCXX_VOLATILE, 3742
quicksort.h, 3742
quiet_NaN
std::numeric_limits< _Tp >, 2575
std::numeric_limits< bool >, 2580
std::numeric_limits< char >, 2586
std::numeric_limits< char16_t >, 2591
std::numeric_limits< char32_t >, 2596
std::numeric_limits< double >, 2601
std::numeric_limits< float >, 2607
std::numeric_limits< int >, 2612
std::numeric_limits< long >, 2617
std::numeric_limits< long double >, 2622
std::numeric_limits< long long >, 2628
std::numeric_limits< short >, 2633
std::numeric_limits< signed char >, 2638
std::numeric_limits< unsigned char >, 2643
std::numeric_limits< unsigned int >, 2649
std::numeric_limits< unsigned long >, 2654
std::numeric_limits< unsigned long long >, 2659
std::numeric_limits< unsigned short >, 2664
std::numeric_limits< wchar_t >, 2670
quoted_string.h, 3432

r_erase_fn_imps.hpp, 3613
radix
std::__numeric_limits_base, 790
std::numeric_limits< _Tp >, 2577
std::numeric_limits< bool >, 2583
std::numeric_limits< char >, 2588
std::numeric_limits< char16_t >, 2593
std::numeric_limits< char32_t >, 2598
std::numeric_limits< double >, 2604
std::numeric_limits< float >, 2609
std::numeric_limits< int >, 2614
std::numeric_limits< long >, 2619
std::numeric_limits< long double >, 2625
std::numeric_limits< long long >, 2630
std::numeric_limits< short >, 2635
std::numeric_limits< signed char >, 2640
std::numeric_limits< unsigned char >, 2646
std::numeric_limits< unsigned int >, 2651
std::numeric_limits< unsigned long >, 2656
std::numeric_limits< unsigned long long >, 2661
std::numeric_limits< unsigned short >, 2667
std::numeric_limits< wchar_t >, 2672
random, 3749
Random Number Distributions, 234
Random Number Generation, 233
generate_canonical, 233
Random Number Generators, 244
minstd_rand, 245
minstd_rand0, 245
mt19937, 245
mt19937_64, 245
operator<<, 246
Random Number Utilities, 246
random.h, 3433
random.tcc, 3435, 3439
random_number.h, 3742
random_sample
SGI, 155, 156
random_sample_n
SGI, 156
random_shuffle
Mutating, 29, 30
random_shuffle.h, 3743
random_shuffle_minimal_n
__gnu_parallel::_Settings, 1013
range_access.h, 3441
ranged_hash_fn.hpp, 3638
ranged_probe_fn.hpp, 3638
ranges, 3749
ranges_algo.h, 3443
ranges_algobase.h, 3447
ranges_base.h, 3447
ranges_cmp.h, 3448
ranges_uninitialized.h, 3448
ranges_util.h, 3448
ratio, 3749–3751
ratio_add

Rational Arithmetic, 355
ratio_divide
Rational Arithmetic, 355
ratio_multiply
Rational Arithmetic, 355
ratio_subtract
Rational Arithmetic, 356
Rational Arithmetic, 354
ratio_add, 355
ratio_divide, 355
ratio_multiply, 355
ratio_subtract, 356
rb_tree, 3660
rb_tree_.hpp, 3646
rbegin
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 846
std, 635
std::basic_string< _CharT, _Traits, _Alloc >, 1622
std::deque< _Tp, _Alloc >, 2049
std::list< _Tp, _Alloc >, 2342, 2343
std::map< _Key, _Tp, _Compare, _Alloc >, 2402, 2403
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2496
std::multiset< _Key, _Compare, _Alloc >, 2521
std::set< _Key, _Compare, _Alloc >, 2949
std::vector< _Tp, _Alloc >, 3305
std::vector< bool, _Alloc >, 3325, 3326
rc.hpp, 3646
rc_binomial_heap_.hpp, 3647
rc_string_base.h, 3661
rdbuf
std::basic_fstream< _CharT, _Traits >, 1146
std::basic_ifstream< _CharT, _Traits >, 1190
std::basic_ios< _CharT, _Traits >, 1215, 1216
std::basic_iostream< _CharT, _Traits >, 1257, 1258
std::basic_istream< _CharT, _Traits >, 1296, 1297
std::basic_istream< _CharT, _Traits >::sentry, 2887
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1335, 1336
std::basic_ofstream< _CharT, _Traits >, 1372
std::basic_ostream< _CharT, _Traits >, 1401, 1402
std::basic_ostream< _CharT, _Traits >::sentry, 2916
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1434
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1703
rdstate
std::basic_fstream< _CharT, _Traits >, 1146
std::basic_ifstream< _CharT, _Traits >, 1190
std::basic_ios< _CharT, _Traits >, 1216
std::basic_iostream< _CharT, _Traits >, 1258
std::basic_istream< _CharT, _Traits >, 1297
std::basic_istream< _CharT, _Traits >::sentry, 2888
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1336
std::basic_ofstream< _CharT, _Traits >, 1372
std::basic_ostream< _CharT, _Traits >, 1402
std::basic_ostream< _CharT, _Traits >::sentry, 2917
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1435
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1703
read
std::basic_fstream< _CharT, _Traits >, 1147
std::basic_ifstream< _CharT, _Traits >, 1191
std::basic_iostream< _CharT, _Traits >, 1258
std::basic_istream< _CharT, _Traits >, 1297
std::basic_istream< _CharT, _Traits >::sentry, 2888
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1336
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1703
readsome
std::basic_fstream< _CharT, _Traits >, 1147
std::basic_ifstream< _CharT, _Traits >, 1191
std::basic_iostream< _CharT, _Traits >, 1259
std::basic_istream< _CharT, _Traits >, 1298
std::basic_istream< _CharT, _Traits >::sentry, 2888
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1337
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1704
ready
std::match_results< _Bi_iter, _Alloc >, 2416
rebind
std::pointer_traits< _Ptr >, 2756
std::pointer_traits< _Tp * >, 2757
reduce
Generalized Numeric operations, 13, 14
ref
std::reference_wrapper< _Tp >, 2815
reference
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1724
__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1729
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1739
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1742
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >, 2306
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_ite
Node, _Alloc >, 2310
std::back_insert_iterator< _Container >, 1077
std::front_insert_iterator< _Container >, 2162

std::insert_iterator< _Container >, 2253
std::istream_iterator< _Tp, _CharT, _Traits, _Dist >, 2292
std::istreambuf_iterator< _CharT, _Traits >, 2295
std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >, 2300
std::ostream_iterator< _Tp, _CharT, _Traits >, 2688
std::ostreambuf_iterator< _CharT, _Traits >, 2692
std::raw_storage_iterator< _OutputIterator, _Tp >, 2797
std::set< _Key, _Compare, _Alloc >, 2933
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3172
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3203
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3231
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3258
refwrap.h, 3449
regex, 3751, 3752
Regular Expressions, 255
regex.h, 3450
regex.tcc, 3452
regex_automaton.h, 3452
regex_automaton.tcc, 3453
regex_compiler.h, 3453
regex_compiler.tcc, 3454
regex_constants.h, 3454
regex_error
std::regex_error, 2816
regex_error.h, 3455
regex_executor.h, 3456
regex_executor.tcc, 3456
regex_iterator
std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >, 2818
regex_match
Regular Expressions, 259–262
regex_replace
Regular Expressions, 263–266
regex_scanner.h, 3457
regex_scanner.tcc, 3457
regex_search
Regular Expressions, 266, 268–270
regex_token_iterator
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >, 2820–2822
regex_traits
std::regex_traits< _Ch_type >, 2825
register_callback
std::basic_fstream< _CharT, _Traits >, 1148
std::basic_ifstream< _CharT, _Traits >, 1192
std::basic_ios< _CharT, _Traits >, 1217
std::basic_iostream< _CharT, _Traits >, 1259
std::basic_istream< _CharT, _Traits >, 1299
std::basic_istream< _CharT, _Traits >::sentry, 2889
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1337
std::basic_ofstream< _CharT, _Traits >, 1373
std::basic_ostream< _CharT, _Traits >, 1403
std::basic_ostream< _CharT, _Traits >::sentry, 2917
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1435
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1705
std::ios_base, 2263
Regular Expressions, 252
cregex_token_iterator, 255
csub_match, 255
operator<<, 256
operator<=>, 256, 257
operator==, 257, 258
regex, 255
regex_match, 259–262
regex_replace, 263–266
regex_search, 266, 268–270
sregex_token_iterator, 255
ssub_match, 255
swap, 271
wcregex_token_iterator, 255
wcsub_match, 255
wregex, 255
wsregex_token_iterator, 255
wssub_match, 255
rehash
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3193
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3222
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3249
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3277
reinterpret_pointer_cast
Pointer Abstractions, 341
release
std::auto_ptr< _Tp >, 1075
std::unique_ptr< _Tp, _Dp >, 3154
std::unique_ptr< _Tp[], _Dp >, 3162
remove
Mutating, 30
std::forward_list< _Tp, _Alloc >, 2153
std::list< _Tp, _Alloc >, 2343
remove_all_extents_t
Metaprogramming, 351
remove_copy
Mutating, 31

remove_copy_if
Mutating, 31
remove_extent_t
Metaprogramming, 352
remove_if
Mutating, 32
std::forward_list< _Tp, _Alloc >, 2154
std::list< _Tp, _Alloc >, 2343
remove_pointer_t
Metaprogramming, 352
remove_reference_t
Metaprogramming, 352
rend
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 847
std, 636
std::basic_string< _CharT, _Traits, _Alloc >, 1623
std::deque< _Tp, _Alloc >, 2049
std::list< _Tp, _Alloc >, 2343
std::map< _Key, _Tp, _Compare, _Alloc >, 2403
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2497
std::multiset< _Key, _Compare, _Alloc >, 2522
std::set< _Key, _Compare, _Alloc >, 2950
std::vector< _Tp, _Alloc >, 3306
std::vector< bool, _Alloc >, 3326
replace
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 847–853
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1527–1536
Mutating, 32
std::basic_string< _CharT, _Traits, _Alloc >, 1623–1635
replace_copy
std, 636
replace_copy_if
Mutating, 33
replace_if
Mutating, 33
replace_minimal_n
__gnu_parallel::_Settings, 1014
requires_hosted.h, 3457
reserve
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 853
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1537, 1538
std::basic_string< _CharT, _Traits, _Alloc >, 1636, 1637
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3193
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3223
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3249
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3277
std::vector< _Tp, _Alloc >, 3306
std::vector< bool, _Alloc >, 3326
reset
std::auto_ptr< _Tp >, 1075
std::bernoulli_distribution, 1719
std::binomial_distribution< _IntType >, 1750
std::bitset< _Nb >, 1768
std::cauchy_distribution< _RealType >, 1776
std::chi_squared_distribution< _RealType >, 1799
std::discrete_distribution< _IntType >, 2063
std::exponential_distribution< _RealType >, 2121
std::extreme_value_distribution< _RealType >, 2124
std::fisher_f_distribution< _RealType >, 2133
std::gamma_distribution< _RealType >, 2181
std::geometric_distribution< _IntType >, 2184
std::lognormal_distribution< _RealType >, 2368
std::negative_binomial_distribution< _IntType >, 2532
std::normal_distribution< _RealType >, 2538
std::piecewise_constant_distribution< _RealType >, 2746
std::piecewise_linear_distribution< _RealType >, 2749
std::poisson_distribution< _IntType >, 2761
std::student_t_distribution< _RealType >, 3033
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2080
std::uniform_int_distribution< _IntType >, 3145
std::uniform_real_distribution< _RealType >, 3148
std::unique_ptr< _Tp, _Dp >, 3154
std::unique_ptr< _Tp[], _Dp >, 3162, 3163
std::weibull_distribution< _RealType >, 3348
resetiosflags
std, 637
resize
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 854
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1539
__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >, 2237
Numeric Arrays, 231
std::basic_string< _CharT, _Traits, _Alloc >, 1637, 1638
std::deque< _Tp, _Alloc >, 2049, 2050
std::forward_list< _Tp, _Alloc >, 2154
std::list< _Tp, _Alloc >, 2344
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2080
std::vector< _Tp, _Alloc >, 3306, 3307
std::vector< bool, _Alloc >, 3326, 3327

resize_fn_imps.hpp, 3633
resize_no_store_hash_fn_imps.hpp, 3633
resize_policy.hpp, 3626
resize_store_hash_fn_imps.hpp, 3633, 3634
result_of_t
    Metaprogramming, 352
result_type
    __gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >, 1733
    __gnu_cxx::equal_to< _Tp >, 2108
    __gnu_cxx::project1st< _Arg1, _Arg2 >, 2770
    __gnu_cxx::project2nd< _Arg1, _Arg2 >, 2771
    __gnu_cxx::select1st< _Pair >, 2862
    __gnu_cxx::select2nd< _Pair >, 2862
    __gnu_cxx::subtractive_rng, 3044
    __gnu_cxx::unary_compose< _Operation1, _Operation2 >, 3138
    __gnu_parallel::_EqualFromLess< _T1, _T2, _Compare >, 876
    __gnu_parallel::_EqualTo< _T1, _T2 >, 877
    __gnu_parallel::_Less< _T1, _T2 >, 903
    __gnu_parallel::_Lexicographic< _T1, _T2, _Compare >, 904
    __gnu_parallel::_LexicographicReverse< _T1, _T2, _Compare >, 905
    __gnu_parallel::_Multiplies< _Tp1, _Tp2, _Result >, 931
    __gnu_parallel::_Plus< _Tp1, _Tp2, _Result >, 943
    __gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >, 741
    __gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >, 742
    __gnu_parallel::__unary_negate< _Predicate, argument_type >, 807
    std::bernoulli_distribution, 1717
    std::binary_function< _Arg1, _Arg2, _Result >, 1735
    std::binary_negate< _Predicate >, 1745
    std::binder1st< _Operation >, 1746
    std::binder2nd< _Operation >, 1748
    std::binomial_distribution< _IntType >, 1749
    std::cauchy_distribution< _RealType >, 1775
    std::chi_squared_distribution< _RealType >, 1798
    std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >, 1882
    std::const_mem_fun1_t< _Ret, _Tp, _Arg >, 1883
    std::const_mem_fun_ref_t< _Ret, _Tp >, 1884
    std::const_mem_fun_t< _Ret, _Tp >, 1885
    std::discard_block_engine< _RandomNumberEngine, __p, __r >, 2057
    std::discrete_distribution< _IntType >, 2062
    std::divides< _Tp >, 2065
    std::divides< void >, 2066
    std::equal_to< _Tp >, 2109
    std::experimental::fundamentals_v2::owner_less< shared_ptr< _Tp > >, 2704
    std::experimental::fundamentals_v2::owner_less< weak_ptr< _Tp > >, 2706
    std::exponential_distribution< _RealType >, 2120
    std::extreme_value_distribution< _RealType >, 2123
    std::fisher_f_distribution< _RealType >, 2132
    std::gamma_distribution< _RealType >, 2180
    std::geometric_distribution< _IntType >, 2183
    std::greater< _Tp >, 2197
    std::greater< void >, 2199
    std::greater_equal< _Tp >, 2200
    std::greater_equal< void >, 2201
    std::hash< __gnu_cxx::throw_value_limit >, 2209
    std::hash< __gnu_cxx::throw_value_random >, 2210
    std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >, 2242
    std::less< _Tp >, 2313
    std::less_equal< _Tp >, 2315
    std::less_equal< void >, 2316
    std::linear_congruential_engine< _UIntType, __a, __c, __m >, 2319
    std::logical_and< _Tp >, 2360
    std::logical_and< void >, 2361
    std::logical_not< _Tp >, 2362
    std::logical_not< void >, 2363
    std::logical_or< _Tp >, 2364
    std::logical_or< void >, 2365
    std::lognormal_distribution< _RealType >, 2367
    std::mem_fun1_ref_t< _Ret, _Tp, _Arg >, 2420
    std::mem_fun1_t< _Ret, _Tp, _Arg >, 2421
    std::mem_fun_ref_t< _Ret, _Tp >, 2422
    std::mem_fun_t< _Ret, _Tp >, 2423
    std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >, 2426
    std::minus< _Tp >, 2435
    std::minus< void >, 2436
    std::modulus< _Tp >, 2438
    std::modulus< void >, 2439
    std::multiplies< _Tp >, 2500
    std::multiplies< void >, 2501
    std::negate< _Tp >, 2528
    std::negate< void >, 2529
    std::negative_binomial_distribution< _IntType >, 2531
    std::normal_distribution< _RealType >, 2537
    std::not_equal_to< _Tp >, 2540
    std::not_equal_to< void >, 2542
    std::owner_less< shared_ptr< _Tp > >, 2705
    std::owner_less< void >, 2706
    std::owner_less< weak_ptr< _Tp > >, 2707

std::piecewise_constant_distribution< _RealType >, 2744
std::piecewise_linear_distribution< _RealType >, 2748
std::plus< _Tp >, 2751
std::pointer_to_binary_function< _Arg1, _Arg2, _Result >, 2753
std::pointer_to_unary_function< _Arg, _Result >, 2755
std::poisson_distribution< _IntType >, 2760
std::random_device, 2782
std::seed_seq, 2861
std::shuffle_order_engine< _RandomNumberEngine, __k >, 2970
std::student_t_distribution< _RealType >, 3032
std::subtract_with_carry_engine< _UIntType, __w, __s, __r >, 3040
std::unary_function< _Arg, _Result >, 3139
std::unary_negate< _Predicate >, 3140
std::uniform_int_distribution< _IntType >, 3144
std::uniform_real_distribution< _RealType >, 3146
std::weibull_distribution< _RealType >, 3347
rethrow_exception
exception_ptr.h, 3399
Exceptions, 130
rethrow_if_nested
Exceptions, 130
rethrow_nested
std::nested_exception, 2533
return_temporary_buffer
std, 637
reverse
Mutating, 34
std::forward_list< _Tp, _Alloc >, 2155
std::list< _Tp, _Alloc >, 2344
reverse_copy
Mutating, 34
reverse_iteration
__gnu_pbds::container_traits< Cntnr >, 1899
reverse_iterator
std::reverse_iterator< _Iterator >, 2835
std::set< _Key, _Compare, _Alloc >, 2933
rfind
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 855, 856
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1539–1543
std::basic_string< _CharT, _Traits, _Alloc >, 1638–1641
riemann_zeta
Mathematical Special Functions, 212
TR1 Mathematical Special Functions, 251
riemann_zetaf
Mathematical Special Functions, 212
riemann_zetal
Mathematical Special Functions, 213
right
std, 637
std::basic_fstream< _CharT, _Traits >, 1158
std::basic_ifstream< _CharT, _Traits >, 1201
std::basic_ios< _CharT, _Traits >, 1224
std::basic_iostream< _CharT, _Traits >, 1270
std::basic_istream< _CharT, _Traits >, 1308
std::basic_istream< _CharT, _Traits >::sentry, 2897
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1347
std::basic_ofstream< _CharT, _Traits >, 1381
std::basic_ostream< _CharT, _Traits >, 1411
std::basic_ostream< _CharT, _Traits >::sentry, 2924
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1445
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1715
std::ios_base, 2269
rope, 3661
ropeimpl.h, 3664
rotate
Mutating, 35
rotate_copy
Mutating, 35
rotate_fn_imps.hpp, 3613
round
Time, 374, 376
round_error
std::numeric_limits< _Tp >, 2575
std::numeric_limits< bool >, 2580
std::numeric_limits< char >, 2586
std::numeric_limits< char16_t >, 2591
std::numeric_limits< char32_t >, 2596
std::numeric_limits< double >, 2601
std::numeric_limits< float >, 2607
std::numeric_limits< int >, 2612
std::numeric_limits< long >, 2617
std::numeric_limits< long double >, 2622
std::numeric_limits< long long >, 2628
std::numeric_limits< short >, 2633
std::numeric_limits< signed char >, 2638
std::numeric_limits< unsigned char >, 2643
std::numeric_limits< unsigned int >, 2649
std::numeric_limits< unsigned long >, 2654
std::numeric_limits< unsigned long long >, 2659
std::numeric_limits< unsigned short >, 2664
std::numeric_limits< wchar_t >, 2670
round_indeterminate
std, 581
round_style
std::__numeric_limits_base, 790
std::numeric_limits< _Tp >, 2577

std::numeric_limits< bool >, 2583
std::numeric_limits< char >, 2588
std::numeric_limits< char16_t >, 2593
std::numeric_limits< char32_t >, 2598
std::numeric_limits< double >, 2604
std::numeric_limits< float >, 2609
std::numeric_limits< int >, 2614
std::numeric_limits< long >, 2619
std::numeric_limits< long double >, 2625
std::numeric_limits< long long >, 2630
std::numeric_limits< short >, 2635
std::numeric_limits< signed char >, 2640
std::numeric_limits< unsigned char >, 2646
std::numeric_limits< unsigned int >, 2651
std::numeric_limits< unsigned long >, 2656
std::numeric_limits< unsigned long long >, 2661
std::numeric_limits< unsigned short >, 2667
std::numeric_limits< wchar_t >, 2672
round_to_nearest
std, 581
round_toward_infinity
std, 581
round_toward_neg_infinity
std, 581
round_toward_zero
std, 581
runtime_error
std::runtime_error, 2844

safe_base.h, 3586
safe_container.h, 3587
safe_iterator.h, 3587
safe_iterator.tcc, 3588
safe_local_iterator.h, 3589
safe_local_iterator.tcc, 3590
safe_sequence.h, 3590
safe_sequence.tcc, 3590
safe_unordered_base.h, 3590
safe_unordered_container.h, 3591
safe_unordered_container.tcc, 3591
sample
std, 638
std::experimental, 702
sample_probe_fn
__gnu_pbds::sample_probe_fn, 2845
sample_probe_fn.hpp, 3639
sample_range_hashing
__gnu_pbds::sample_range_hashing, 2846
__gnu_pbds::sample_resize_policy, 2850
__gnu_pbds::sample_resize_trigger, 2853
__gnu_pbds::sample_size_policy, 2854
sample_range_hashing.hpp, 3639
sample_ranged_hash_fn
__gnu_pbds::sample_ranged_hash_fn, 2847
sample_ranged_hash_fn.hpp, 3639
sample_ranged_probe_fn.hpp, 3640
sample_resize_policy
__gnu_pbds::sample_resize_policy, 2849
sample_resize_policy.hpp, 3648
sample_resize_trigger
__gnu_pbds::sample_resize_trigger, 2851
sample_resize_trigger.hpp, 3648
sample_size_policy
__gnu_pbds::sample_size_policy, 2854
sample_size_policy.hpp, 3648
sample_tree_node_update.hpp, 3651
sample_trie_access_traits.hpp, 3652
sample_trie_node_update
__gnu_pbds::sample_trie_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >, 2856
sample_trie_node_update.hpp, 3652
sample_update_policy
__gnu_pbds::sample_update_policy, 2857
sample_update_policy.hpp, 3642
Sampling Distributions, 242
sat_arith.h, 3457
sbumpc
__gnu_cxx::enc_filebuf< _CharT >, 2093
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 3000
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3018
std::basic_filebuf< _CharT, _Traits >, 1101
std::basic_streambuf< _CharT, _Traits >, 1466
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1660
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3337
scan_is
std::__ctype_abstract_base< _CharT >, 758
std::ctype< _CharT >, 1919
std::ctype< char >, 1947, 1948
std::ctype< wchar_t >, 1973
std::ctype_byname< _CharT >, 1987
std::ctype_byname< char >, 2006, 2007
scan_not
std::__ctype_abstract_base< _CharT >, 759
std::ctype< _CharT >, 1920
std::ctype< char >, 1948, 1950
std::ctype< wchar_t >, 1973
std::ctype_byname< _CharT >, 1987
std::ctype_byname< char >, 2007, 2008
scientific
std, 638
std::basic_fstream< _CharT, _Traits >, 1159
std::basic_ifstream< _CharT, _Traits >, 1201
std::basic_ios< _CharT, _Traits >, 1224
std::basic_iostream< _CharT, _Traits >, 1270
std::basic_istream< _CharT, _Traits >, 1308
std::basic_istream< _CharT, _Traits >::sentry, 2897

std::basic_istringstream< _CharT, _Traits, _Alloc >, 1347
std::basic_ofstream< _CharT, _Traits >, 1381
std::basic_ostream< _CharT, _Traits >, 1411
std::basic_ostream< _CharT, _Traits >::sentry, 2924
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1445
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1716
std::ios_base, 2269
scoped_allocator, 3752
search
Non-Mutating, 57, 58
std, 638
std::__parallel, 674
search.h, 3743
search_minimal_n
__gnu_parallel::_Settings, 1014
search_n
Non-Mutating, 58, 59
second
__gnu_cxx::pair< _T1, _T2 >, 2711
__gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >, 899
std::pair< _T1, _T2 >, 2715
second_argument_type
__gnu_cxx::equal_to< _Tp >, 2108
__gnu_cxx::project1st< _Arg1, _Arg2 >, 2770
__gnu_cxx::project2nd< _Arg1, _Arg2 >, 2771
__gnu_parallel::_EqualFromLess< _T1, _T2, _Compare >, 876
__gnu_parallel::_EqualTo< _T1, _T2 >, 877
__gnu_parallel::_Less< _T1, _T2 >, 903
__gnu_parallel::_Lexicographic< _T1, _T2, _Compare >, 904
__gnu_parallel::_LexicographicReverse< _T1, _T2, _Compare >, 906
__gnu_parallel::_Multiplies< _Tp1, _Tp2, _Result >, 931
__gnu_parallel::_Plus< _Tp1, _Tp2, _Result >, 943
std::binary_function< _Arg1, _Arg2, _Result >, 1735
std::binary_negate< _Predicate >, 1745
std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >, 1882
std::const_mem_fun1_t< _Ret, _Tp, _Arg >, 1883
std::divides< _Tp >, 2065
std::divides< void >, 2066
std::equal_to< _Tp >, 2109
std::experimental::fundamentals_v2::owner_less< shared_ptr< _Tp > >, 2704
std::experimental::fundamentals_v2::owner_less< weak_ptr< _Tp > >, 2706
std::greater< _Tp >, 2197
std::greater< void >, 2199
std::greater_equal< _Tp >, 2200
std::greater_equal< void >, 2201
std::less< _Tp >, 2313
std::less_equal< _Tp >, 2315
std::less_equal< void >, 2316
std::logical_and< _Tp >, 2360
std::logical_and< void >, 2362
std::logical_or< _Tp >, 2364
std::logical_or< void >, 2366
std::mem_fun1_ref_t< _Ret, _Tp, _Arg >, 2420
std::mem_fun1_t< _Ret, _Tp, _Arg >, 2421
std::minus< _Tp >, 2435
std::minus< void >, 2436
std::modulus< _Tp >, 2438
std::modulus< void >, 2439
std::multiplies< _Tp >, 2500
std::multiplies< void >, 2501
std::not_equal_to< _Tp >, 2541
std::not_equal_to< void >, 2542
std::owner_less< shared_ptr< _Tp > >, 2705
std::owner_less< void >, 2706
std::owner_less< weak_ptr< _Tp > >, 2707
std::plus< _Tp >, 2751
std::pointer_to_binary_function< _Arg1, _Arg2, _Result >, 2753
second_type
__gnu_cxx::pair< _T1, _T2 >, 2710
__gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >, 896
std::pair< _T1, _T2 >, 2713
seconds
Time, 365
seed
std::discard_block_engine< _RandomNumberEngine, __p, __r >, 2059
std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >, 2244
std::linear_congruential_engine< _UIntType, __a, __c, __m >, 2320
std::shuffle_order_engine< _RandomNumberEngine, __k >, 2972
std::subtract_with_carry_engine< _UIntType, __w, __s, __r >, 3041
seed_seq
std::seed_seq, 2861
seekdir
std::basic_fstream< _CharT, _Traits >, 1118
std::basic_ifstream< _CharT, _Traits >, 1169
std::basic_ios< _CharT, _Traits >, 1208
std::basic_iostream< _CharT, _Traits >, 1231
std::basic_istream< _CharT, _Traits >, 1278
std::basic_istream< _CharT, _Traits >::sentry, 2870
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1316

std::basic_ofstream< _CharT, _Traits >, 1354
std::basic_ostream< _CharT, _Traits >, 1388
std::basic_ostream< _CharT, _Traits >::sentry, 2903
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1420
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1676
std::ios_base, 2260
seekg
std::basic_fstream< _CharT, _Traits >, 1148
std::basic_ifstream< _CharT, _Traits >, 1192
std::basic_iostream< _CharT, _Traits >, 1260
std::basic_istream< _CharT, _Traits >, 1299
std::basic_istream< _CharT, _Traits >::sentry, 2889, 2890
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1338
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1705
seekoff
__gnu_cxx::enc_filebuf< _CharT >, 2093
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 3000
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3018
std::basic_filebuf< _CharT, _Traits >, 1101
std::basic_streambuf< _CharT, _Traits >, 1466
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1660
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3337
seekp
std::basic_fstream< _CharT, _Traits >, 1149
std::basic_iostream< _CharT, _Traits >, 1261
std::basic_ofstream< _CharT, _Traits >, 1373
std::basic_ostream< _CharT, _Traits >, 1403
std::basic_ostream< _CharT, _Traits >::sentry, 2917, 2918
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1435, 1436
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1706
seekpos
__gnu_cxx::enc_filebuf< _CharT >, 2094
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 3001
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3018
std::basic_filebuf< _CharT, _Traits >, 1101
std::basic_streambuf< _CharT, _Traits >, 1466
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1660
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3338
select_on_container_copy_construction
__gnu_cxx::__alloc_traits< _Alloc, typename >, 738
std::allocator_traits< _Alloc >, 1036
std::allocator_traits< allocator< _Tp > >, 1045
std::allocator_traits< allocator< void > >, 1053
std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1062
semaphore, 3753
semaphore_base.h, 3457
sentry
std::basic_istream< _CharT, _Traits >::sentry, 2870
std::basic_ostream< _CharT, _Traits >::sentry, 2904
Sequences, 121
sequential
__gnu_parallel, 423
set, 3753, 3754
__gnu_parallel::_Settings, 1010
std::bitset< _Nb >, 1768, 1769
std::set< _Key, _Compare, _Alloc >, 2934–2936
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2080, 2081
Set Operations, 92
includes, 93
set_difference, 94
set_intersection, 95, 96
set_symmetric_difference, 96, 97
set_union, 98
set.h, 3591
set_default_resource
memory_resource, 3707
set_difference
Set Operations, 94
set_difference_minimal_n
__gnu_parallel::_Settings, 1014
set_intersection
Set Operations, 95, 96
set_intersection_minimal_n
__gnu_parallel::_Settings, 1014
set_load
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1780
set_loads
__gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >, 2226
__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >, 2237
set_new_handler
std, 638
set_num_threads
__gnu_parallel::balanced_quicksort_tag, 1086
__gnu_parallel::balanced_tag, 1087
__gnu_parallel::default_parallel_tag, 2023
__gnu_parallel::exact_tag, 2115
__gnu_parallel::multiway_mergesort_exact_tag, 2524
__gnu_parallel::multiway_mergesort_sampling_tag, 2525
__gnu_parallel::multiway_mergesort_tag, 2526
__gnu_parallel::omp_loop_static_tag, 2683

__gnu_parallel::omp_loop_tag, 2684
__gnu_parallel::parallel_tag, 2720
__gnu_parallel::quicksort_tag, 2779
__gnu_parallel::sampling_tag, 2859
__gnu_parallel::unbalanced_tag, 3141
set_operations.h, 3744
set_symmetric_difference
Set Operations, 96, 97
set_symmetric_difference_minimal_n
__gnu_parallel::_Settings, 1014
set_terminate
Exceptions, 130
set_unexpected
Exceptions, 131
set_union
Set Operations, 98
set_union_minimal_n
__gnu_parallel::_Settings, 1014
setbase
std, 638
setbuf
__gnu_cxx::enc_filebuf< _CharT >, 2094
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 3001
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3019
std::basic_filebuf< _CharT, _Traits >, 1102
std::basic_streambuf< _CharT, _Traits >, 1466
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1661
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3338
setf
std::basic_fstream< _CharT, _Traits >, 1150
std::basic_ifstream< _CharT, _Traits >, 1193
std::basic_ios< _CharT, _Traits >, 1217
std::basic_iostream< _CharT, _Traits >, 1261, 1262
std::basic_istream< _CharT, _Traits >, 1300
std::basic_istream< _CharT, _Traits >::sentry, 2890
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1339
std::basic_ofstream< _CharT, _Traits >, 1374
std::basic_ostream< _CharT, _Traits >, 1404
std::basic_ostream< _CharT, _Traits >::sentry, 2918
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1436
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1706, 1707
std::ios_base, 2264
setfill
std, 639
setg
__gnu_cxx::enc_filebuf< _CharT >, 2094
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 3001
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3019
std::basic_filebuf< _CharT, _Traits >, 1102
std::basic_streambuf< _CharT, _Traits >, 1467
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1661
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3338
setiosflags
std, 639
setp
__gnu_cxx::enc_filebuf< _CharT >, 2095
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 3002
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3019
std::basic_filebuf< _CharT, _Traits >, 1103
std::basic_streambuf< _CharT, _Traits >, 1467
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1662
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3339
setprecision
std, 639
setstate
std::basic_fstream< _CharT, _Traits >, 1150
std::basic_ifstream< _CharT, _Traits >, 1194
std::basic_ios< _CharT, _Traits >, 1218
std::basic_iostream< _CharT, _Traits >, 1262
std::basic_istream< _CharT, _Traits >, 1300
std::basic_istream< _CharT, _Traits >::sentry, 2891
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1339
std::basic_ofstream< _CharT, _Traits >, 1374
std::basic_ostream< _CharT, _Traits >, 1404
std::basic_ostream< _CharT, _Traits >::sentry, 2919
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1437
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1707
settings.h, 3744
_GLIBCXX_PARALLEL_CONDITION, 3745
setw
std, 639
sgetc
__gnu_cxx::enc_filebuf< _CharT >, 2095
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 3002
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3020
std::basic_filebuf< _CharT, _Traits >, 1103
std::basic_streambuf< _CharT, _Traits >, 1468
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1662
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3339
sgetn
__gnu_cxx::enc_filebuf< _CharT >, 2095
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 3002
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3020
std::basic_filebuf< _CharT, _Traits >, 1103
std::basic_streambuf< _CharT, _Traits >, 1468
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1662
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3339

SGI, 147
_Find_first, 152
_Find_next, 152
_Unchecked_flip, 152
_Unchecked_reset, 152
_Unchecked_set, 153
_Unchecked_test, 153
__median, 150
compose1, 153
compose2, 153
constant0, 153
constant1, 153
constant2, 154
copy_n, 154
distance, 154
identity_element, 154
lexicographical_compare_3way, 155
power, 155
random_sample, 155, 156
random_sample_n, 156
uninitialized_copy_n, 156
shared_future
std::shared_future< _Res >, 2953
std::shared_future< _Res & >, 2955
std::shared_future< void >, 2957
shared_mutex, 3754
shared_ptr
std::shared_ptr< _Tp >, 2961–2967
shared_ptr.h, 3457, 3458
shared_ptr_atomic.h, 3460
shared_ptr_base.h, 3462
shift
Numeric Arrays, 231
showbase
std, 641
std::basic_fstream< _CharT, _Traits >, 1159
std::basic_ifstream< _CharT, _Traits >, 1201
std::basic_ios< _CharT, _Traits >, 1224
std::basic_iostream< _CharT, _Traits >, 1270
std::basic_istream< _CharT, _Traits >, 1308
std::basic_istream< _CharT, _Traits >::sentry, 2897
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1347
std::basic_ofstream< _CharT, _Traits >, 1381
std::basic_ostream< _CharT, _Traits >, 1411
std::basic_ostream< _CharT, _Traits >::sentry, 2924
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1445
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1716
std::ios_base, 2269
showmanyc
__gnu_cxx::enc_filebuf< _CharT >, 2096
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 3003
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3020
std::basic_filebuf< _CharT, _Traits >, 1103
std::basic_streambuf< _CharT, _Traits >, 1468
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1663
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3340
showpoint
std, 641
std::basic_fstream< _CharT, _Traits >, 1159
std::basic_ifstream< _CharT, _Traits >, 1201
std::basic_ios< _CharT, _Traits >, 1224
std::basic_iostream< _CharT, _Traits >, 1270
std::basic_istream< _CharT, _Traits >, 1308
std::basic_istream< _CharT, _Traits >::sentry, 2897
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1347
std::basic_ofstream< _CharT, _Traits >, 1381
std::basic_ostream< _CharT, _Traits >, 1411
std::basic_ostream< _CharT, _Traits >::sentry, 2925
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1445
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1716
std::ios_base, 2269
showpos
std, 641
std::basic_fstream< _CharT, _Traits >, 1159
std::basic_ifstream< _CharT, _Traits >, 1201
std::basic_ios< _CharT, _Traits >, 1224
std::basic_iostream< _CharT, _Traits >, 1270
std::basic_istream< _CharT, _Traits >, 1308
std::basic_istream< _CharT, _Traits >::sentry, 2897
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1347
std::basic_ofstream< _CharT, _Traits >, 1381
std::basic_ostream< _CharT, _Traits >, 1411
std::basic_ostream< _CharT, _Traits >::sentry, 2925
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1445
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1716
std::ios_base, 2269
shrink_to_fit
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 856
std::basic_string< _CharT, _Traits, _Alloc >, 1642
std::deque< _Tp, _Alloc >, 2050
std::vector< _Tp, _Alloc >, 3307
std::vector< bool, _Alloc >, 3327
shuffle
Mutating, 36
shuffle_order_engine
std::shuffle_order_engine< _RandomNumberEngine, __k >, 2970, 2971

signaling_NaN
std::numeric_limits< _Tp >, 2575
std::numeric_limits< bool >, 2581
std::numeric_limits< char >, 2586
std::numeric_limits< char16_t >, 2591
std::numeric_limits< char32_t >, 2596
std::numeric_limits< double >, 2602
std::numeric_limits< float >, 2607
std::numeric_limits< int >, 2612
std::numeric_limits< long >, 2617
std::numeric_limits< long double >, 2623
std::numeric_limits< long long >, 2628
std::numeric_limits< short >, 2633
std::numeric_limits< signed char >, 2638
std::numeric_limits< unsigned char >, 2644
std::numeric_limits< unsigned int >, 2649
std::numeric_limits< unsigned long >, 2654
std::numeric_limits< unsigned long long >, 2659
std::numeric_limits< unsigned short >, 2665
std::numeric_limits< wchar_t >, 2670
simd, 3605
simd_abi::deduce< _Tp, _Np,... >, 2019
sin
Complex Numbers, 186
math.h, 3702
sinh
Complex Numbers, 187
math.h, 3702
size
__gnu_cxx::_Temporary_buffer< _ForwardIterator, _Tp >, 1019
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 856
__gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >, 3049
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1544
Numeric Arrays, 232
std, 641
std::_Temporary_buffer< _ForwardIterator, _Tp >, 1021
std::basic_string< _CharT, _Traits, _Alloc >, 1642
std::bitset< _Nb >, 1769
std::deque< _Tp, _Alloc >, 2050
std::list< _Tp, _Alloc >, 2344
std::map< _Key, _Tp, _Compare, _Alloc >, 2403
std::match_results< _Bi_iter, _Alloc >, 2416
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2497
std::multiset< _Key, _Compare, _Alloc >, 2522
std::priority_queue< _Tp, _Sequence, _Compare >, 2769
std::queue< _Tp, _Sequence >, 2778
std::set< _Key, _Compare, _Alloc >, 2950
std::stack< _Tp, _Sequence >, 2987
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2081
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3193
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3223
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3249
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3278
std::vector< _Tp, _Alloc >, 3307
std::vector< bool, _Alloc >, 3327
size_fn_imps.hpp, 3634
size_type
__gnu_pbds::hash_prime_size_policy, 2232
__gnu_pbds::sample_range_hashing, 2846
__gnu_pbds::sample_resize_policy, 2849
__gnu_pbds::sample_resize_trigger, 2851
__gnu_pbds::sample_size_policy, 2854
__gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >, 3116
std::allocator_traits< _Alloc >, 1033
std::allocator_traits< allocator< _Tp > >, 1040
std::allocator_traits< allocator< void > >, 1048, 1049
std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1057
std::set< _Key, _Compare, _Alloc >, 2933
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3173
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3203
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3232
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3258
skipws
std, 641
std::basic_fstream< _CharT, _Traits >, 1159
std::basic_ifstream< _CharT, _Traits >, 1201
std::basic_ios< _CharT, _Traits >, 1224
std::basic_iostream< _CharT, _Traits >, 1270
std::basic_istream< _CharT, _Traits >, 1308
std::basic_istream< _CharT, _Traits >::sentry, 2897
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1347
std::basic_ofstream< _CharT, _Traits >, 1382
std::basic_ostream< _CharT, _Traits >, 1412
std::basic_ostream< _CharT, _Traits >::sentry, 2925
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1445
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1716
std::ios_base, 2269

sleep_for
std::this_thread, 721
sleep_until
std::this_thread, 721
slice
Numeric Arrays, 219
slice_array
Numeric Arrays, 220
slice_array.h, 3463
slist, 3671
snextc
__gnu_cxx::enc_filebuf< _CharT >, 2096
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 3003
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3021
std::basic_filebuf< _CharT, _Traits >, 1104
std::basic_streambuf< _CharT, _Traits >, 1469
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1663
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3340
sort
Sorting, 79
std::forward_list< _Tp, _Alloc >, 2155
std::list< _Tp, _Alloc >, 2344, 2345
sort.h, 3745
sort_heap
Heap, 91
sort_minimal_n
__gnu_parallel::_Settings, 1014
sort_mwms_oversampling
__gnu_parallel::_Settings, 1014
sort_qs_num_samples_preset
__gnu_parallel::_Settings, 1014
sort_qsb_base_case_maximal_n
__gnu_parallel::_Settings, 1015
Sorting, 60
clamp, 62, 63
inplace_merge, 63, 64
is_sorted, 64, 65
is_sorted_until, 65
lexicographical_compare, 66
lexicographical_compare_three_way, 67
max, 67, 68
max_element, 68, 69
merge, 69, 70
min, 70, 71
min_element, 72
minmax, 72, 73
minmax_element, 73
next_permutation, 74
nth_element, 75
partial_sort, 76
partial_sort_copy, 77
prev_permutation, 78
sort, 79
stable_sort, 80
source_location, 3755
span, 3755
specfun.h, 3463
sph_bessel
Mathematical Special Functions, 213
TR1 Mathematical Special Functions, 251
sph_besself
Mathematical Special Functions, 213
sph_bessell
Mathematical Special Functions, 214
sph_legendre
Mathematical Special Functions, 214
TR1 Mathematical Special Functions, 251
sph_legendref
Mathematical Special Functions, 214
sph_legendrel
Mathematical Special Functions, 215
sph_neumann
Mathematical Special Functions, 215
TR1 Mathematical Special Functions, 251
sph_neumannf
Mathematical Special Functions, 215
sph_neumannl
Mathematical Special Functions, 215
splay_fn_imps.hpp, 3649
splay_tree_.hpp, 3649
splice
std::list< _Tp, _Alloc >, 2345, 2346
splice_after
std::forward_list< _Tp, _Alloc >, 2155, 2156
split_fn_imps.hpp, 3645
split_join_can_throw
__gnu_pbds::container_traits< Cntnr >, 1899
split_join_fn_imps.hpp, 3627
sputbackc
__gnu_cxx::enc_filebuf< _CharT >, 2096
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 3003
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3021
std::basic_filebuf< _CharT, _Traits >, 1104
std::basic_streambuf< _CharT, _Traits >, 1469
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1663
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3340
sputc
__gnu_cxx::enc_filebuf< _CharT >, 2097
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 3004
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3021
std::basic_filebuf< _CharT, _Traits >, 1104
std::basic_streambuf< _CharT, _Traits >, 1469
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1664
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3341
sputn

__gnu_cxx::enc_filebuf< _CharT >, 2097
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 3004
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3023
std::basic_filebuf< _CharT, _Traits >, 1105
std::basic_streambuf< _CharT, _Traits >, 1470
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1664
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3341
sqrt
Complex Numbers, 187
math.h, 3703
sregex_token_iterator
Regular Expressions, 255
sso_string_base.h, 3672
sstream, 3755
sstream.tcc, 3466
ssub_match
Regular Expressions, 255
stable_partition
Mutating, 36
stable_sort
Sorting, 80
stack, 3756
std::stack< _Tp, _Sequence >, 2987
standard_policies.hpp, 3649
start
Numeric Arrays, 232
state
std::fpos< _StateT >, 2159
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3342
std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >, 3351
static_pointer_cast
Pointer Abstractions, 342
std, 642
std, 464
_Construct, 587
_Destroy, 587
_Destroy_n, 587
__find_if_not, 581
__find_if_not_n, 581
__gcd, 581
__gen_two_uniform_ints, 581
__inplace_stable_sort, 582
__lg, 582
__merge_adaptive, 582
__merge_without_buffer, 583
__move_median_to_first, 583
__move_merge, 583
__move_merge_adaptive, 583
__move_merge_adaptive_backward, 584
__partition, 584
__ptr_rebind, 578
__reverse, 584
__rotate, 585
__rotate_adaptive, 585
__sample, 585, 586
__search_n_aux, 586
__stable_partition_adaptive, 586
__umap_traits, 578
__ummap_traits, 579
__umset_traits, 579
__uset_traits, 579
acosh, 587
advance, 587
begin, 588
boolalpha, 589
cbegin, 589
cend, 589
cerr, 646
chars_format, 580
cin, 646
clog, 646
compare_three_way_result_t, 579
const_pointer_cast, 589
cout, 646
crbegin, 589
crend, 590
data, 590, 591
dec, 591
default_sentinel, 646
defaultfloat, 591
denorm_absent, 580
denorm_indeterminate, 580
denorm_present, 580
destroying_delete, 646
distance, 591
dynamic_pointer_cast, 592
empty, 592
end, 592, 593
endl, 593
ends, 593
fabs, 593
fixed, 593
float_denorm_style, 580
float_round_style, 580
flush, 594
frexp, 594
from_chars, 594
get, 594–596
get_money, 596
get_new_handler, 596
get_temporary_buffer, 596
get_time, 596
getline, 597, 598
hex, 599
hexfloat, 599
ignore, 646

index_sequence, 579
index_sequence_for, 579
internal, 599
io_errc, 581
isalnum, 599
isalpha, 599
isblank, 600
iscntrl, 600
isdigit, 600
isgraph, 600
islower, 600
isprint, 600
ispunct, 601
isspace, 601
isupper, 601
isxdigit, 601
left, 601
make_index_sequence, 579
make_integer_sequence, 579
new_handler, 579
noboolalpha, 601
noshowbase, 602
noshowpoint, 602
noshowpos, 602
noskipws, 602
nounitbuf, 602
nouppercase, 602
oct, 602
operator!=, 602, 603
operator<, 605–607
operator<<, 608–617
operator<=, 617, 618
operator<=>, 618–620
operator>, 626, 627
operator>>, 627–632
operator>=, 627
operator+, 604, 605
operator==, 621–626
operator&, 603
operator^, 632
operator|, 633
put_money, 633
put_time, 633
rbegin, 635
rend, 636
replace_copy, 636
resetiosflags, 637
return_temporary_buffer, 637
right, 637
round_indeterminate, 581
round_to_nearest, 581
round_toward_infinity, 581
round_toward_neg_infinity, 581
round_toward_zero, 581
sample, 638
scientific, 638
search, 638
set_new_handler, 638
setbase, 638
setfill, 639
setiosflags, 639
setprecision, 639
setw, 639
showbase, 641
showpoint, 641
showpos, 641
size, 641
skipws, 641
static_pointer_cast, 642
streamoff, 580
streampos, 580
streamsize, 580
swap, 642–644
tolower, 645
toupper, 645
u16streampos, 580
u32streampos, 580
unitbuf, 645
uppercase, 645
wcerr, 646
wcin, 646
wclog, 647
wcout, 647
ws, 645
wstreampos, 580
std::__basic_future< _Res >, 738
_M_get_result, 740
std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >, 742
do_out, 743
in, 744
out, 744
unshift, 745
std::__ctype_abstract_base< _CharT >, 749
char_type, 751
do_is, 751
do_narrow, 752
do_scan_is, 753
do_scan_not, 753
do_tolower, 754
do_toupper, 755
do_widen, 756
is, 757
narrow, 757, 758
scan_is, 758
scan_not, 759
tolower, 759, 760
toupper, 760

widen, 761
std::__cxx11::collate< _CharT >, 1856
~collate, 1858
char_type, 1857
collate, 1857, 1858
compare, 1858
do_compare, 1858
do_hash, 1859
do_transform, 1859
hash, 1860
id, 1861
string_type, 1857
transform, 1860
std::__cxx11::collate_byname< _CharT >, 1861
char_type, 1862
string_type, 1862
std::__debug, 647
swap, 654
std::__debug::bitset< _Nb >, 1759
std::__debug::deque< _Tp, _Allocator >, 2026
std::__debug::forward_list< _Tp, _Alloc >, 2136
std::__debug::list< _Tp, _Allocator >, 2324
std::__debug::map< _Key, _Tp, _Compare, _Allocator >, 2377
std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >, 2473
std::__debug::multiset< _Key, _Compare, _Allocator >, 2501
std::__debug::set< _Key, _Compare, _Allocator >, 2926
std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3164
std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3195
std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3224
std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3250
std::__debug::vector< _Tp, _Allocator >, 3286
vector, 3289
std::__detail, 654
__from_chars_alnum, 656
__from_chars_pow2_base, 656
operator<<, 656
operator>>, 656
std::__detail::_BracketMatcher< _TraitsT, __icase, __collate >, 864
std::__detail::_Compiler< _TraitsT >, 867
std::__detail::_Executor< _BiIter, _Alloc, _TraitsT, __dfs_mode >, 877
std::__detail::_List_node_base, 910
std::__detail::_List_node_header, 911
std::__detail::_Quoted_string< _String, _CharT >, 951
std::__detail::_Scanner< _CharT >, 1007
_TokenT, 1008
std::__detail::_StateSeq< _TraitsT >, 1017
std::__fwdlist::_Iterator< _Const, _Ptr >, 892
operator==, 893
std::__fwdlist::_Node< _ValPtr >, 931
std::__fwdlist::_Node_base< _VoidPtr >, 934
std::__is_fast_hash< _Hash >, 776
std::__is_location_invariant< _Tp >, 776
std::__new_allocator< _Tp >, 784
std::__numeric_limits_base, 786
digits, 788
digits10, 788
has_denorm, 788
has_denorm_loss, 788
has_infinity, 788
has_quiet_NaN, 789
has_signaling_NaN, 789
is_bounded, 789
is_exact, 789
is_iec559, 789
is_integer, 789
is_modulo, 789
is_signed, 789
is_specialized, 789
max_digits10, 789
max_exponent, 790
max_exponent10, 790
min_exponent, 790
min_exponent10, 790
radix, 790
round_style, 790
tinyness_before, 790
traps, 790
std::__parallel, 657
search, 674
std::__parallel::_CRandNumber< _MustBeInt >, 868
std::__unspecified__::exception_ptr, 2117
std::_Base_bitset< 0 >, 859
_M_w, 861
std::_Base_bitset< 1 >, 861
_M_w, 863
std::_Base_bitset< _Nw >, 858
_M_w, 859
std::_Bind< _Signature >, 863
std::_Bind_result< _Result, _Signature >, 863
std::_Deque_base< _Tp, _Alloc >, 868
_M_initialize_map, 869
std::_Deque_iterator< _Tp, _Ref, _Ptr >, 870
_M_set_node, 871
std::_Function_base, 879
std::_Fwd_list_base< _Tp, _Alloc >, 880
std::_Fwd_list_const_iterator< _Tp >, 882
operator==, 883
std::_Fwd_list_iterator< _Tp >, 883
operator==, 884

std::_Fwd_list_node< _Tp >, 884
std::_Fwd_list_node_base, 885
std::_List_base< _Tp, _Alloc >, 906
std::_List_const_iterator< _Tp >, 907
std::_List_iterator< _Tp >, 908
std::_List_node< _Tp >, 909
std::_Not_fn< _Fn >, 940
std::_Placeholder< _Num >, 942
std::_Sp_ebo_helper< _Nm, _Tp, false >, 1015
std::_Sp_ebo_helper< _Nm, _Tp, true >, 1016
std::_Temporary_buffer< _ForwardIterator, _Tp >, 1020
    _M_requested_size, 1021
    _Temporary_buffer, 1021
    begin, 1021
    end, 1021
    size, 1021
std::_Vector_base< _Tp, _Alloc >, 1022
std::add_const< _Tp >, 1023
std::add_cv< _Tp >, 1024
std::add_lvalue_reference< _Tp >, 1024
std::add_pointer< _Tp >, 1024
std::add_rvalue_reference< _Tp >, 1024
std::add_volatile< _Tp >, 1025
std::adopt_lock_t, 1025
std::aligned_storage< _Len, _Align >, 1025
std::aligned_union< _Len, _Types >, 1026
std::alignment_of< _Tp >, 1026
std::allocator< _Tp >, 1029
std::allocator_traits< _Alloc >, 1030
    allocate, 1034
    allocator_type, 1032
    const_pointer, 1032
    const_void_pointer, 1032
    construct, 1034
    deallocate, 1035
    destroy, 1035
    difference_type, 1033
    is_always_equal, 1033
    max_size, 1036
    pointer, 1033
    propagate_on_container_copy_assignment, 1033
    propagate_on_container_move_assignment, 1033
    propagate_on_container_swap, 1033
    select_on_container_copy_construction, 1036
    size_type, 1033
    value_type, 1034
    void_pointer, 1034
std::allocator_traits< allocator< _Tp > >, 1036
    allocate, 1041
    allocator_type, 1038
    const_pointer, 1038
    const_void_pointer, 1038
    construct, 1042
    deallocate, 1043
    destroy, 1043, 1044
    difference_type, 1038, 1039
    is_always_equal, 1039
    max_size, 1044
    pointer, 1039
    propagate_on_container_copy_assignment, 1039
    propagate_on_container_move_assignment, 1039
    propagate_on_container_swap, 1040
    select_on_container_copy_construction, 1045
    size_type, 1040
    value_type, 1040
    void_pointer, 1040
std::allocator_traits< allocator< void > >, 1045
    allocate, 1049, 1051
    allocator_type, 1047
    const_pointer, 1047
    const_void_pointer, 1047
    construct, 1051
    deallocate, 1051, 1052
    destroy, 1052
    difference_type, 1047
    is_always_equal, 1047, 1048
    max_size, 1053
    pointer, 1048
    propagate_on_container_copy_assignment, 1048
    propagate_on_container_move_assignment, 1048
    propagate_on_container_swap, 1048
    select_on_container_copy_construction, 1053
    size_type, 1048, 1049
    value_type, 1049
    void_pointer, 1049
std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1054
    allocate, 1058, 1059
    allocator_type, 1055
    const_pointer, 1055
    const_void_pointer, 1056
    construct, 1059, 1060
    deallocate, 1060
    destroy, 1061
    difference_type, 1056
    is_always_equal, 1056
    max_size, 1061, 1062
    pointer, 1056
    propagate_on_container_copy_assignment, 1056, 1057
    propagate_on_container_move_assignment, 1057
    propagate_on_container_swap, 1057
    select_on_container_copy_construction, 1062
    size_type, 1057
    value_type, 1057, 1058
    void_pointer, 1058
std::array< _Tp, _Nm >, 1066
std::atomic< _Tp >, 1068

std::atomic< _Tp * >, 1069
std::atomic_flag, 1071
std::auto_ptr< _Tp >, 1071
  ~auto_ptr, 1073
  auto_ptr, 1073
  element_type, 1072
  get, 1074
  operator->, 1074
  operator=, 1074, 1075
  operator*, 1074
  release, 1075
  reset, 1075
std::auto_ptr_ref< _Tp1 >, 1076
std::back_insert_iterator< _Container >, 1076
  back_insert_iterator, 1078
  container_type, 1077
  iterator_category, 1077
  operator++, 1078
  operator=, 1078
  operator*, 1078
  pointer, 1077
  reference, 1077
  value_type, 1077
std::bad_alloc, 1079
  what, 1079
std::bad_cast, 1080
  what, 1081
std::bad_exception, 1081
  what, 1082
std::bad_function_call, 1082
  what, 1083
std::bad_typeid, 1084
  what, 1084
std::bad_weak_ptr, 1084
  what, 1085
std::basic_filebuf< _CharT, _Traits >, 1089
  _M_buf, 1108
  _M_buf_locale, 1108
  _M_buf_size, 1108
  _M_create_pback, 1093
  _M_destroy_pback, 1093
  _M_ext_buf, 1108
  _M_ext_buf_size, 1108
  _M_ext_next, 1108
  _M_in_beg, 1108
  _M_in_cur, 1108
  _M_in_end, 1109
  _M_mode, 1109
  _M_out_beg, 1109
  _M_out_cur, 1109
  _M_out_end, 1109
  _M_pback, 1109
  _M_pback_cur_save, 1109
  _M_pback_end_save, 1110
  _M_pback_init, 1110
  _M_reading, 1110
  _M_set_buffer, 1093
  ~basic_filebuf, 1093
  basic_filebuf, 1093
  close, 1094
  eback, 1094
  egptr, 1094
  epptr, 1094
  gbump, 1095
  getloc, 1095
  gptr, 1095
  imbue, 1096
  in_avail, 1096
  is_open, 1096
  open, 1096, 1097
  overflow, 1098
  pbackfail, 1098
  pbase, 1099
  pbump, 1099
  pptr, 1099
  pubimbue, 1100
  pubseekoff, 1100
  pubseekpos, 1100
  pubsetbuf, 1101
  pubsync, 1101
  sbumpc, 1101
  seekoff, 1101
  seekpos, 1101
  setbuf, 1102
  setg, 1102
  setp, 1103
  sgetc, 1103
  sgetn, 1103
  showmanyc, 1103
  snextc, 1104
  sputbackc, 1104
  sputc, 1104
  sputn, 1105
  sungetc, 1105
  sync, 1105
  uflow, 1106
  underflow, 1106
  xsgetn, 1106
  xsputn, 1107
std::basic_fstream< _CharT, _Traits >, 1110
  _M_gcount, 1155
  _M_getloc, 1120
  ~basic_fstream, 1119
  adjustfield, 1155
  app, 1155
  ate, 1155
  bad, 1120
  badbit, 1155

basefield, 1156
basic_fstream, 1118, 1119
beg, 1156
binary, 1156
boolalpha, 1156
clear, 1120
close, 1120
copyfmt, 1120
cur, 1156
dec, 1156
end, 1156
eof, 1122
eofbit, 1157
event, 1118
event_callback, 1117
exceptions, 1122
fail, 1123
failbit, 1157
fill, 1123
fixed, 1157
flags, 1123, 1124
floatfield, 1157
flush, 1124
gcount, 1124
get, 1124–1126
getline, 1127, 1128
getloc, 1128
good, 1128
goodbit, 1157
hex, 1158
ignore, 1128, 1129
imbue, 1129
in, 1158
init, 1130
internal, 1158
iostate, 1117
is_open, 1130
iword, 1130
left, 1158
narrow, 1131
oct, 1158
open, 1131, 1132
openmode, 1118
operator bool, 1132
operator!, 1132
operator<<, 1132–1134, 1136–1138
operator>>, 1138–1143
out, 1158
peek, 1144
precision, 1144
put, 1144
putback, 1145
pword, 1145
rdbuf, 1146
rdstate, 1146
read, 1147
readsome, 1147
register_callback, 1148
right, 1158
scientific, 1159
seekdir, 1118
seekg, 1148
seekp, 1149
setf, 1150
setstate, 1150
showbase, 1159
showpoint, 1159
showpos, 1159
skipws, 1159
sync, 1151
sync_with_stdio, 1151
tellg, 1151
tellp, 1152
tie, 1152
trunc, 1159
unget, 1153
unitbuf, 1159
unsetf, 1153
uppercase, 1159
widen, 1153
width, 1154
write, 1154
xalloc, 1155
std::basic_ifstream< _CharT, _Traits >, 1162
_M_gcount, 1197
_M_getloc, 1171
__num_put_type, 1168
~basic_ifstream, 1170
adjustfield, 1197
app, 1198
ate, 1198
bad, 1171
badbit, 1198
basefield, 1198
basic_ifstream, 1169, 1170
beg, 1198
binary, 1198
boolalpha, 1198
clear, 1171
close, 1171
copyfmt, 1171
cur, 1199
dec, 1199
end, 1199
eof, 1173
eofbit, 1199
event, 1169
event_callback, 1168

exceptions, 1173
fail, 1174
failbit, 1199
fill, 1174
fixed, 1199
flags, 1174, 1175
floatfield, 1200
gcount, 1175
get, 1175–1177
getline, 1178, 1179
getloc, 1179
good, 1179
goodbit, 1200
hex, 1200
ignore, 1179, 1180
imbue, 1180
in, 1200
init, 1181
internal, 1200
iostate, 1168
is_open, 1181
iword, 1181
left, 1200
narrow, 1181
oct, 1201
open, 1182
openmode, 1169
operator bool, 1183
operator!, 1183
operator>>, 1183–1188
out, 1201
peek, 1188
precision, 1188, 1189
putback, 1189
pword, 1189
rdbuf, 1190
rdstate, 1190
read, 1191
readsome, 1191
register_callback, 1192
right, 1201
scientific, 1201
seekdir, 1169
seekg, 1192
setf, 1193
setstate, 1194
showbase, 1201
showpoint, 1201
showpos, 1201
skipws, 1201
sync, 1194
sync_with_stdio, 1194
tellg, 1195
tie, 1195
trunc, 1201
unget, 1196
unitbuf, 1202
unsetf, 1196
uppercase, 1202
widen, 1196
width, 1197
xalloc, 1197
std::basic_ios< _CharT, _Traits >, 1203
_M_getloc, 1209
__ctype_type, 1207
__num_get_type, 1207
__num_put_type, 1207
~basic_ios, 1209
adjustfield, 1220
app, 1220
ate, 1220
bad, 1209
badbit, 1221
basefield, 1221
basic_ios, 1209
beg, 1221
binary, 1221
boolalpha, 1221
char_type, 1207
clear, 1210
copyfmt, 1210
cur, 1221
dec, 1221
end, 1222
eof, 1210
eofbit, 1222
event, 1209
event_callback, 1207
exceptions, 1211
fail, 1211
failbit, 1222
fill, 1212
fixed, 1222
flags, 1212
floatfield, 1222
getloc, 1213
good, 1213
goodbit, 1222
hex, 1223
imbue, 1213
in, 1223
init, 1213
int_type, 1207
internal, 1223
iostate, 1207
iword, 1214
left, 1223
narrow, 1214

oct, 1223
off_type, 1208
openmode, 1208
operator bool, 1214
operator!, 1215
out, 1223
pos_type, 1208
precision, 1215
pword, 1215
rdbuf, 1215, 1216
rdstate, 1216
register_callback, 1217
right, 1224
scientific, 1224
seekdir, 1208
setf, 1217
setstate, 1218
showbase, 1224
showpoint, 1224
showpos, 1224
skipws, 1224
sync_with_stdio, 1218
tie, 1218
traits_type, 1208
trunc, 1224
unitbuf, 1224
unsetf, 1219
uppercase, 1224
widen, 1219
width, 1219, 1220
xalloc, 1220
std::basic_iostream< _CharT, _Traits >, 1225
_M_gcount, 1267
_M_getloc, 1232
~basic_iostream, 1232
adjustfield, 1267
app, 1267
ate, 1267
bad, 1232
badbit, 1267
basefield, 1267
basic_iostream, 1232
beg, 1267
binary, 1267
boolalpha, 1268
clear, 1233
copyfmt, 1233
cur, 1268
dec, 1268
end, 1268
eof, 1233
eofbit, 1268
event, 1232
event_callback, 1231
exceptions, 1233, 1234
fail, 1234
failbit, 1268
fill, 1234, 1235
fixed, 1269
flags, 1235
floatfield, 1269
flush, 1235
gcount, 1236
get, 1236, 1237, 1239
getline, 1239, 1240
getloc, 1240
good, 1241
goodbit, 1269
hex, 1269
ignore, 1241
imbue, 1243
in, 1269
init, 1243
internal, 1269
iostate, 1231
iword, 1243
left, 1270
narrow, 1244
oct, 1270
openmode, 1231
operator bool, 1244
operator!, 1244
operator<<, 1244–1249
operator>>, 1250–1255
out, 1270
peek, 1255
precision, 1255, 1256
put, 1256
putback, 1256
pword, 1257
rdbuf, 1257, 1258
rdstate, 1258
read, 1258
readsome, 1259
register_callback, 1259
right, 1270
scientific, 1270
seekdir, 1231
seekg, 1260
seekp, 1261
setf, 1261, 1262
setstate, 1262
showbase, 1270
showpoint, 1270
showpos, 1270
skipws, 1270
sync, 1262
sync_with_stdio, 1263

tellg, 1263
tellp, 1263
tie, 1264
trunc, 1271
unget, 1264
unitbuf, 1271
unsetf, 1265
uppercase, 1271
widen, 1265
width, 1265
write, 1266
xalloc, 1266
std::basic_istream< _CharT, _Traits >, 1271
_M_gcount, 1304
_M_getloc, 1278
__num_put_type, 1277
~basic_istream, 1278
adjustfield, 1304
app, 1304
ate, 1304
bad, 1279
badbit, 1305
basefield, 1305
basic_istream, 1278
beg, 1305
binary, 1305
boolalpha, 1305
clear, 1279
copyfmt, 1279
cur, 1305
dec, 1305
end, 1306
eof, 1279
eofbit, 1306
event, 1278
event_callback, 1277
exceptions, 1280
fail, 1280
failbit, 1306
fill, 1281
fixed, 1306
flags, 1281
floatfield, 1306
gcount, 1282
get, 1282–1284
getline, 1284, 1285
getloc, 1285
good, 1286
goodbit, 1306
hex, 1307
ignore, 1286
imbue, 1288
in, 1307
init, 1288
internal, 1307
iostate, 1277
iword, 1288
left, 1307
narrow, 1289
oct, 1307
openmode, 1277
operator bool, 1289
operator!, 1289
operator>>, 1289–1294
out, 1307
peek, 1295
precision, 1295
putback, 1295
pword, 1296
rdbuf, 1296, 1297
rdstate, 1297
read, 1297
readsome, 1298
register_callback, 1299
right, 1308
scientific, 1308
seekdir, 1278
seekg, 1299
setf, 1300
setstate, 1300
showbase, 1308
showpoint, 1308
showpos, 1308
skipws, 1308
sync, 1301
sync_with_stdio, 1301
tellg, 1301
tie, 1302
trunc, 1308
unget, 1302
unitbuf, 1308
unsetf, 1303
uppercase, 1308
widen, 1303
width, 1303, 1304
xalloc, 1304
std::basic_istream< _CharT, _Traits >::sentry, 2863
_M_gcount, 2894
_M_getloc, 2871
__num_put_type, 2869
~basic_istream, 2871
adjustfield, 2894
app, 2894
ate, 2894
bad, 2871
badbit, 2894
basefield, 2894
basic_istream, 2871

beg, 2894
binary, 2894
boolalpha, 2894
clear, 2871
copyfmt, 2871
cur, 2895
dec, 2895
end, 2895
eof, 2872
eofbit, 2895
event, 2870
event_callback, 2869
exceptions, 2872
fail, 2873
failbit, 2895
fill, 2873
fixed, 2895
flags, 2873
floatfield, 2895
fmtflags, 2895
gcount, 2874
get, 2874, 2875
getline, 2876, 2878
getloc, 2878
good, 2878
goodbit, 2896
hex, 2896
ignore, 2878, 2879
imbue, 2879
in, 2896
init, 2880
internal, 2896
iostate, 2869
iword, 2880
left, 2896
narrow, 2880
oct, 2896
openmode, 2869
operator bool, 2881
operator!, 2881
operator>>, 2881–2886
out, 2896
peek, 2886
precision, 2886
putback, 2886
pword, 2887
rdbuf, 2887
rdstate, 2888
read, 2888
readsome, 2888
register_callback, 2889
right, 2897
scientific, 2897
seekdir, 2870
seekg, 2889, 2890
sentry, 2870
setf, 2890
setstate, 2891
showbase, 2897
showpoint, 2897
showpos, 2897
skipws, 2897
sync, 2891
sync_with_stdio, 2891
tellg, 2891
tie, 2892
traits_type, 2870
trunc, 2897
unget, 2892
unitbuf, 2897
unsetf, 2892
uppercase, 2897
widen, 2893
width, 2893
xalloc, 2893
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1309
_M_gcount, 1343
_M_getloc, 1318
__num_put_type, 1314
~basic_istringstream, 1317
adjustfield, 1343
app, 1344
ate, 1344
bad, 1318
badbit, 1344
basefield, 1344
basic_istringstream, 1317
beg, 1344
binary, 1344
boolalpha, 1344
clear, 1318
copyfmt, 1318
cur, 1345
dec, 1345
end, 1345
eof, 1319
eofbit, 1345
event, 1317
event_callback, 1314
exceptions, 1319
fail, 1320
failbit, 1345
fill, 1320
fixed, 1345
flags, 1320, 1321
floatfield, 1346
gcount, 1321
get, 1321–1323

getline, 1324
getloc, 1325
good, 1325
goodbit, 1346
hex, 1346
ignore, 1325
imbue, 1326
in, 1346
init, 1326
internal, 1346
iostate, 1316
iword, 1326
left, 1346
narrow, 1327
oct, 1347
openmode, 1316
operator bool, 1327
operator!, 1327
operator>>, 1328, 1329, 1331–1333
out, 1347
peek, 1334
precision, 1334
putback, 1334
pword, 1335
rdbuf, 1335, 1336
rdstate, 1336
read, 1336
readsome, 1337
register_callback, 1337
right, 1347
scientific, 1347
seekdir, 1316
seekg, 1338
setf, 1339
setstate, 1339
showbase, 1347
showpoint, 1347
showpos, 1347
skipws, 1347
str, 1340
sync, 1340
sync_with_stdio, 1340
tellg, 1341
tie, 1341
trunc, 1347
unget, 1342
unitbuf, 1348
unsetf, 1342
uppercase, 1348
widen, 1342
width, 1343
xalloc, 1343
std::basic_ofstream< _CharT, _Traits >, 1348
_M_getloc, 1356
__num_get_type, 1353
~basic_ofstream, 1355
adjustfield, 1378
app, 1378
ate, 1378
bad, 1356
badbit, 1378
basefield, 1378
basic_ofstream, 1354, 1355
beg, 1378
binary, 1379
boolalpha, 1379
clear, 1356
close, 1356
copyfmt, 1356
cur, 1379
dec, 1379
end, 1379
eof, 1358
eofbit, 1379
event, 1354
event_callback, 1353
exceptions, 1358
fail, 1359
failbit, 1379
fill, 1359
fixed, 1380
flags, 1359, 1360
floatfield, 1380
flush, 1360
getloc, 1360
good, 1360
goodbit, 1380
hex, 1380
imbue, 1361
in, 1380
init, 1361
internal, 1381
iostate, 1353
is_open, 1361
iword, 1361
left, 1381
narrow, 1363
oct, 1381
open, 1363, 1364
openmode, 1354
operator bool, 1364
operator!, 1364
operator<<, 1364–1366, 1368–1370
out, 1381
precision, 1370, 1371
put, 1371
pword, 1371
rdbuf, 1372

rdstate, 1372
register_callback, 1373
right, 1381
scientific, 1381
seekdir, 1354
seekp, 1373
setf, 1374
setstate, 1374
showbase, 1381
showpoint, 1381
showpos, 1381
skipws, 1382
sync_with_stdio, 1375
tellp, 1375
tie, 1375
trunc, 1382
unitbuf, 1382
unsetf, 1376
uppercase, 1382
widen, 1376
width, 1376, 1377
write, 1377
xalloc, 1377
std::basic_ostream< _CharT, _Traits >, 1382
_M_getloc, 1389
__num_get_type, 1387
~basic_ostream, 1389
adjustfield, 1408
app, 1408
ate, 1408
bad, 1389
badbit, 1408
basefield, 1408
basic_ostream, 1389
beg, 1408
binary, 1409
boolalpha, 1409
clear, 1389
copyfmt, 1390
cur, 1409
dec, 1409
end, 1409
eof, 1390
eofbit, 1409
event, 1389
event_callback, 1387
exceptions, 1390, 1391
fail, 1391
failbit, 1409
fill, 1391, 1392
fixed, 1410
flags, 1392
floatfield, 1410
flush, 1392
getloc, 1393
good, 1393
goodbit, 1410
hex, 1410
imbue, 1393
in, 1410
init, 1394
internal, 1411
iostate, 1388
iword, 1394
left, 1411
narrow, 1394
oct, 1411
openmode, 1388
operator bool, 1395
operator!, 1395
operator<<, 1395–1400
out, 1411
precision, 1400
put, 1401
pword, 1401
rdbuf, 1401, 1402
rdstate, 1402
register_callback, 1403
right, 1411
scientific, 1411
seekdir, 1388
seekp, 1403
setf, 1404
setstate, 1404
showbase, 1411
showpoint, 1411
showpos, 1411
skipws, 1412
sync_with_stdio, 1405
tellp, 1405
tie, 1405
trunc, 1412
unitbuf, 1412
unsetf, 1406
uppercase, 1412
widen, 1406
width, 1406, 1407
write, 1407
xalloc, 1407
std::basic_ostream< _CharT, _Traits >::sentry, 2898
_M_getloc, 2904
__num_get_type, 2902
~basic_ostream, 2904
~sentry, 2904
adjustfield, 2922
app, 2922
ate, 2922
bad, 2904

badbit, 2922
basefield, 2922
basic_ostream, 2904
beg, 2922
binary, 2922
boolalpha, 2922
clear, 2905
copyfmt, 2905
cur, 2922
dec, 2922
end, 2922
eof, 2905
eofbit, 2923
event, 2903
event_callback, 2902
exceptions, 2905
fail, 2906
failbit, 2923
fill, 2906
fixed, 2923
flags, 2907
floatfield, 2923
flush, 2907
fmtflags, 2923
getloc, 2907
good, 2907
goodbit, 2924
hex, 2924
imbue, 2908
in, 2924
init, 2908
internal, 2924
iostate, 2902
iword, 2908
left, 2924
narrow, 2908
oct, 2924
openmode, 2903
operator bool, 2909
operator!, 2909
operator<<, 2909–2914
out, 2924
precision, 2914
put, 2916
pword, 2916
rdbuf, 2916
rdstate, 2917
register_callback, 2917
right, 2924
scientific, 2924
seekdir, 2903
seekp, 2917, 2918
sentry, 2904
setf, 2918
setstate, 2919
showbase, 2924
showpoint, 2925
showpos, 2925
skipws, 2925
sync_with_stdio, 2919
tellp, 2919
tie, 2919
trunc, 2925
unitbuf, 2925
unsetf, 2920
uppercase, 2925
widen, 2920
width, 2920
write, 2921
xalloc, 2921
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1412
_M_getloc, 1421
__num_get_type, 1419
~basic_ostringstream, 1421
adjustfield, 1442
app, 1442
ate, 1442
bad, 1421
badbit, 1442
basefield, 1442
basic_ostringstream, 1420
beg, 1442
binary, 1442
boolalpha, 1442
clear, 1421
copyfmt, 1423
cur, 1443
dec, 1443
end, 1443
eof, 1423
eofbit, 1443
event, 1420
event_callback, 1419
exceptions, 1423
fail, 1424
failbit, 1443
fill, 1424
fixed, 1443
flags, 1425
floatfield, 1444
flush, 1425
getloc, 1425
good, 1426
goodbit, 1444
hex, 1444
imbue, 1426
in, 1444
init, 1426

internal, 1444
iostate, 1419
iword, 1426
left, 1444
narrow, 1427
oct, 1445
openmode, 1419
operator bool, 1427
operator!, 1427
operator<<, 1428–1432
out, 1445
precision, 1433
put, 1433
pword, 1434
rdbuf, 1434
rdstate, 1435
register_callback, 1435
right, 1445
scientific, 1445
seekdir, 1420
seekp, 1435, 1436
setf, 1436
setstate, 1437
showbase, 1445
showpoint, 1445
showpos, 1445
skipws, 1445
str, 1437
sync_with_stdio, 1437
tellp, 1439
tie, 1439
trunc, 1445
unitbuf, 1446
unsetf, 1439
uppercase, 1446
widen, 1440
width, 1440
write, 1441
xalloc, 1441
std::basic_regex< _Ch_type, _Rx_traits >, 1446
~basic_regex, 1450
assign, 1451–1453
basic_regex, 1448–1450
flags, 1453
getloc, 1453
imbue, 1453
mark_count, 1453
operator=, 1454
swap, 1455
std::basic_streambuf< _CharT, _Traits >, 1455
_M_buf_locale, 1472
_M_in_beg, 1472
_M_in_cur, 1473
_M_in_end, 1473
_M_out_beg, 1473
_M_out_cur, 1473
_M_out_end, 1473
__streambuf_type, 1459
~basic_streambuf, 1460
basic_streambuf, 1460
char_type, 1459
eback, 1460
egptr, 1460
epptr, 1461
gbump, 1461
getloc, 1461
gptr, 1462
imbue, 1462
in_avail, 1462
int_type, 1459
off_type, 1459
overflow, 1462
pbackfail, 1463
pbase, 1463
pbump, 1464
pos_type, 1459
pptr, 1464
pubimbue, 1464
pubseekoff, 1465
pubseekpos, 1465
pubsetbuf, 1465
pubsync, 1465
sbumpc, 1466
seekoff, 1466
seekpos, 1466
setbuf, 1466
setg, 1467
setp, 1467
sgetc, 1468
sgetn, 1468
showmanyc, 1468
snextc, 1469
sputbackc, 1469
sputc, 1469
sputn, 1470
sungetc, 1470
sync, 1470
traits_type, 1460
uflow, 1471
underflow, 1471
xsgetn, 1471
xsputn, 1472
std::basic_string< _CharT, _Traits, _Alloc >, 1545
__resize_and_overwrite, 1561
~basic_string, 1557, 1560
append, 1561–1566
assign, 1566–1572
at, 1572, 1573

back, 1574
basic_string, 1554–1560
begin, 1574
c_str, 1575
capacity, 1575
cbegin, 1575
cend, 1576
clear, 1576
compare, 1576–1581
copy, 1582
crbegin, 1583
crend, 1583
data, 1583, 1584
empty, 1584
end, 1584, 1585
erase, 1585, 1587, 1588
find, 1588–1591
find_first_not_of, 1592–1594
find_first_of, 1595–1597
find_last_not_of, 1598–1600
find_last_of, 1601–1603
front, 1604
get_allocator, 1604
insert, 1604–1611, 1613, 1614
length, 1614, 1615
max_size, 1615
npos, 1644
operator+=, 1615–1617
operator=, 1617–1620
operator[], 1620, 1621
pop_back, 1621
push_back, 1621, 1622
rbegin, 1622
rend, 1623
replace, 1623–1635
reserve, 1636, 1637
resize, 1637, 1638
rfind, 1638–1641
shrink_to_fit, 1642
size, 1642
substr, 1642, 1643
swap, 1643
std::basic_string_view< _CharT, _Traits >, 1644
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1650
_M_buf_locale, 1667
_M_in_beg, 1667
_M_in_cur, 1667
_M_in_end, 1668
_M_mode, 1668
_M_out_beg, 1668
_M_out_cur, 1668
_M_out_end, 1668
basic_stringbuf, 1653
eback, 1654
egptr, 1654
epptr, 1654
gbump, 1654
getloc, 1656
gptr, 1656
imbue, 1656
in_avail, 1657
overflow, 1657
pbackfail, 1657
pbase, 1658
pbump, 1658
pptr, 1658
pubimbue, 1659
pubseekoff, 1659
pubseekpos, 1659
pubsetbuf, 1660
pubsync, 1660
sbumpc, 1660
seekoff, 1660
seekpos, 1660
setbuf, 1661
setg, 1661
setp, 1662
sgetc, 1662
sgetn, 1662
showmanyc, 1663
snextc, 1663
sputbackc, 1663
sputc, 1664
sputn, 1664
str, 1664, 1665
sungetc, 1665
sync, 1665
uflow, 1665
underflow, 1666
xsgetn, 1666
xsputn, 1667
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1668
_M_gcount, 1712
_M_getloc, 1678
~basic_stringstream, 1677
adjustfield, 1712
app, 1712
ate, 1712
bad, 1678
badbit, 1712
basefield, 1713
basic_stringstream, 1677
beg, 1713
binary, 1713
boolalpha, 1713
clear, 1678
copyfmt, 1678
cur, 1713

dec, 1713
end, 1713
eof, 1679
eofbit, 1714
event, 1677
event_callback, 1675
exceptions, 1679
fail, 1680
failbit, 1714
fill, 1680
fixed, 1714
flags, 1680, 1681
floatfield, 1714
flush, 1681
gcount, 1681
get, 1681–1683
getline, 1684, 1685
getloc, 1685
good, 1685
goodbit, 1714
hex, 1715
ignore, 1685, 1686
imbue, 1686
in, 1715
init, 1687
internal, 1715
iostate, 1676
iword, 1687
left, 1715
narrow, 1687
oct, 1715
openmode, 1676
operator bool, 1688
operator!, 1688
operator<<, 1688, 1689, 1691–1694
operator>>, 1694, 1695, 1697–1700
out, 1715
peek, 1701
precision, 1701
put, 1701
putback, 1702
pword, 1702
rdbuf, 1703
rdstate, 1703
read, 1703
readsome, 1704
register_callback, 1705
right, 1715
scientific, 1716
seekdir, 1676
seekg, 1705
seekp, 1706
setf, 1706, 1707
setstate, 1707
showbase, 1716
showpoint, 1716
showpos, 1716
skipws, 1716
str, 1707, 1708
sync, 1708
sync_with_stdio, 1708
tellg, 1709
tellp, 1709
tie, 1709
trunc, 1716
unget, 1710
unitbuf, 1716
unsetf, 1710
uppercase, 1716
widen, 1710
width, 1711
write, 1711
xalloc, 1712
std::bernoulli_distribution, 1717
bernoulli_distribution, 1718
max, 1718
min, 1718
operator(), 1718
operator==, 1719
p, 1718
param, 1718
reset, 1719
result_type, 1717
std::bernoulli_distribution::param_type, 2720
std::bidirectional_iterator_tag, 1719
std::binary_function< _Arg1, _Arg2, _Result >, 1733
first_argument_type, 1735
result_type, 1735
second_argument_type, 1735
std::binary_negate< _Predicate >, 1744
first_argument_type, 1745
result_type, 1745
second_argument_type, 1745
std::binder1st< _Operation >, 1745
argument_type, 1746
result_type, 1746
std::binder2nd< _Operation >, 1747
argument_type, 1748
result_type, 1748
std::binomial_distribution< _IntType >, 1748
max, 1749
min, 1749
operator<<, 1750
operator>>, 1752
operator(), 1749
operator==, 1752
p, 1750
param, 1750

reset, 1750
result_type, 1749
t, 1750
std::binomial_distribution< _IntType >::param_type, 2721
std::bitset< _Nb >, 1760
all, 1765
any, 1765
bitset, 1763, 1764
count, 1765
flip, 1765
none, 1765
operator<<, 1766
operator<<=, 1766
operator>>, 1766
operator>>=, 1766
operator==, 1766
operator&=, 1766
operator[], 1767
operator~, 1768
operator^=, 1767
operator|=, 1768
reset, 1768
set, 1768, 1769
size, 1769
test, 1769
to_string, 1769
to_ulong, 1770
std::bitset< _Nb >::reference, 2813
std::cauchy_distribution< _RealType >, 1774
max, 1775
min, 1775
operator(), 1775
operator==, 1776
param, 1775
reset, 1776
result_type, 1775
std::cauchy_distribution< _RealType >::param_type, 2721
std::char_traits< __gnu_cxx::character< _Value, _Int, _St > >, 1792
std::char_traits< _CharT >, 1791
std::char_traits< wchar_t >, 1794
std::chi_squared_distribution< _RealType >, 1797
max, 1798
min, 1798
operator<<, 1799
operator>>, 1799
operator(), 1798
operator==, 1799
param, 1798
reset, 1799
result_type, 1798
std::chi_squared_distribution< _RealType >::param_type, 2722
std::chrono, 674
std::chrono::duration< _Rep, _Period >, 2068
std::chrono::duration_values< _Rep >, 2069
std::chrono::gps_clock, 2196
std::chrono::hh_mm_ss< _Duration >, 2238
std::chrono::steady_clock, 3026
std::chrono::system_clock, 3045
std::chrono::tai_clock, 3046
std::chrono::time_point< _Clock, _Dur >, 3083
std::chrono::treat_as_floating_point< _Rep >, 3093
std::chrono::tzdb_list, 3136
erase_after, 3137
front, 3137
std::chrono::tzdb_list::const_iterator, 1880
std::chrono::utc_clock, 3280
std::codecvt< _InternT, _ExternT, _StateT >, 1800
do_always_noconv, 1801
do_encoding, 1801
do_in, 1802
do_length, 1802
do_max_length, 1802
do_out, 1802
do_unshift, 1802
in, 1803
out, 1803
unshift, 1804
std::codecvt< _InternT, _ExternT, encoding_state >, 1805
do_always_noconv, 1807
do_encoding, 1807
do_in, 1807, 1808
do_length, 1808
do_max_length, 1808
do_out, 1808, 1809
do_unshift, 1809
in, 1809, 1810
out, 1811, 1812
unshift, 1812, 1813
std::codecvt< char, char, mbstate_t >, 1814
do_always_noconv, 1816
do_encoding, 1816
do_in, 1816
do_length, 1817
do_max_length, 1817
do_out, 1817
do_unshift, 1818
in, 1818, 1819
out, 1820
unshift, 1821, 1822
std::codecvt< char16_t, char, mbstate_t >, 1822
do_always_noconv, 1825
do_encoding, 1825
do_in, 1825
do_length, 1825, 1826
do_max_length, 1826

do_out, 1826
do_unshift, 1827
in, 1827, 1828
out, 1828, 1829
unshift, 1830
std::codecvt< char32_t, char, mbstate_t >, 1831
do_always_noconv, 1834
do_encoding, 1834
do_in, 1834
do_length, 1834, 1835
do_max_length, 1835
do_out, 1835
do_unshift, 1836
in, 1836, 1837
out, 1837, 1838
unshift, 1839
std::codecvt< wchar_t, char, mbstate_t >, 1840
do_always_noconv, 1843
do_encoding, 1843
do_in, 1843
do_length, 1844
do_max_length, 1844
do_out, 1844
do_unshift, 1845
in, 1845, 1846
out, 1847
unshift, 1848, 1849
std::codecvt_base, 1849
std::codecvt_byname< _InternT, _ExternT, _StateT >, 1850
do_always_noconv, 1852
do_encoding, 1852
do_in, 1852
do_length, 1853
do_max_length, 1853
do_out, 1853
do_unshift, 1853
in, 1854
out, 1854
unshift, 1855
std::common_iterator< _It, _Sent >, 1862
std::common_type< _Tp >, 1863
std::common_type< chrono::duration< _Rep, _Period > >, 1863
std::common_type< chrono::duration< _Rep, _Period >, chrono::duration< _Rep, _Period > >, 1864
std::common_type< chrono::duration< _Rep1, _Period1 >, chrono::duration< _Rep2, _Period2 > >, 1864
std::common_type< chrono::time_point< _Clock, _Duration > >, 1864
std::common_type< chrono::time_point< _Clock, _Duration >, chrono::time_point< _Clock, _Duration > >, 1864
std::common_type< chrono::time_point< _Clock, _Duration1 >, chrono::time_point< _Clock, _Duration2 > >, 1865
std::compare_three_way_result< _Tp, _Up >, 1865
std::complex< _Tp >, 1865
complex, 1867
operator+=, 1867
operator-=, 1867
value_type, 1866
std::complex< double >, 1867
complex, 1869
operator+=, 1869
operator-=, 1869
operator/=, 1869, 1870
operator=, 1870
operator*=, 1869
value_type, 1868
std::complex< float >, 1870
complex, 1871
operator+=, 1872
operator-=, 1872
operator/=, 1872
operator=, 1872, 1873
operator*=, 1872
value_type, 1871
std::complex< long double >, 1873
complex, 1874
operator+=, 1875
operator-=, 1875
operator/=, 1875
operator=, 1875
operator*=, 1874
value_type, 1874
std::condition_variable, 1877
std::condition_variable_any, 1878
std::conditional< _Cond, _Iftrue, _Iffalse >, 1879
std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >, 1881
first_argument_type, 1882
result_type, 1882
second_argument_type, 1882
std::const_mem_fun1_t< _Ret, _Tp, _Arg >, 1882
first_argument_type, 1883
result_type, 1883
second_argument_type, 1883
std::const_mem_fun_ref_t< _Ret, _Tp >, 1883
argument_type, 1884
result_type, 1884
std::const_mem_fun_t< _Ret, _Tp >, 1884
argument_type, 1885
result_type, 1885
std::contiguous_iterator_tag, 1903
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>, 1904
copyable_function, 1906, 1907

operator bool, 1907
operator(), 1907
operator=, 1907, 1908
operator==, 1908
swap, 1908
std::counted_iterator< _It >, 1909
std::ctype< _CharT >, 1910
do_is, 1912
do_narrow, 1913
do_scan_is, 1914
do_scan_not, 1914
do_tolower, 1915
do_toupper, 1916
do_widen, 1917
id, 1923
is, 1917, 1918
narrow, 1918, 1919
scan_is, 1919
scan_not, 1920
tolower, 1920, 1921
toupper, 1921
widen, 1922
std::ctype< char >, 1923
~ctype, 1927
char_type, 1926
classic_table, 1927
ctype, 1926
do_is, 1927, 1928
do_narrow, 1929, 1930, 1932
do_scan_is, 1933
do_scan_not, 1934
do_tolower, 1935–1937
do_toupper, 1937–1939
do_widen, 1940, 1941
id, 1958
is, 1942–1944
narrow, 1944–1946
scan_is, 1947, 1948
scan_not, 1948, 1950
table, 1951
table_size, 1958
tolower, 1951–1953
toupper, 1953–1955
widen, 1955–1957
std::ctype< wchar_t >, 1958
~ctype, 1961
char_type, 1960
ctype, 1960, 1961
do_is, 1961, 1962
do_narrow, 1963, 1964
do_scan_is, 1964, 1965
do_scan_not, 1965, 1966
do_tolower, 1966–1968
do_toupper, 1968, 1969
do_widen, 1970, 1971
id, 1976
is, 1971, 1972
narrow, 1972, 1973
scan_is, 1973
scan_not, 1973
tolower, 1974
toupper, 1975
widen, 1975, 1976
std::ctype_base, 1976
std::ctype_byname< _CharT >, 1977
do_is, 1980
do_narrow, 1980, 1981
do_scan_is, 1981
do_scan_not, 1982
do_tolower, 1982, 1983
do_toupper, 1983, 1984
do_widen, 1984
id, 1990
is, 1985
narrow, 1986
scan_is, 1987
scan_not, 1987
tolower, 1988
toupper, 1988, 1989
widen, 1989, 1990
std::ctype_byname< char >, 1990
classic_table, 1993
do_is, 1993, 1994
do_narrow, 1995, 1996
do_scan_is, 1997
do_scan_not, 1998
do_tolower, 1999, 2000
do_toupper, 2000, 2001
do_widen, 2002, 2003
id, 2013
is, 2003, 2004
narrow, 2005, 2006
scan_is, 2006, 2007
scan_not, 2007, 2008
table, 2008
table_size, 2013
tolower, 2008, 2009
toupper, 2010, 2011
widen, 2011, 2012
std::decay< _Tp >, 2014
std::decimal, 681
decimal32_to_long_long, 690
std::decimal::decimal128, 2014
decimal128, 2015
std::decimal::decimal32, 2016
decimal32, 2017
std::decimal::decimal64, 2017
decimal64, 2018

std::default_delete< _Tp >, 2019
default_delete, 2020
operator(), 2020
std::default_delete< _Tp[]>, 2020
default_delete, 2021
operator(), 2021
std::default_sentinel_t, 2024
std::defer_lock_t, 2026
std::deque< _Tp, _Alloc >, 2028
_M_fill_initialize, 2036
_M_new_elements_at_back, 2036
_M_new_elements_at_front, 2036
_M_pop_back_aux, 2037
_M_pop_front_aux, 2037
_M_push_back_aux, 2037
_M_push_front_aux, 2037
_M_range_check, 2037
_M_range_initialize, 2037, 2038
_M_reallocate_map, 2038
_M_reserve_elements_at_back, 2038
_M_reserve_elements_at_front, 2039
_M_reserve_map_at_back, 2039
_M_reserve_map_at_front, 2039
~deque, 2036
assign, 2039, 2040
at, 2040, 2041
back, 2041
begin, 2041
cbegin, 2042
cend, 2042
clear, 2042
crbegin, 2042
crend, 2042
deque, 2033–2035
emplace, 2042
empty, 2043
end, 2043
erase, 2043
front, 2044
get_allocator, 2044
insert, 2044–2046
max_size, 2046
operator=, 2046, 2047
operator[], 2047, 2048
pop_back, 2048
pop_front, 2048
push_back, 2048
push_front, 2049
rbegin, 2049
rend, 2049
resize, 2049, 2050
shrink_to_fit, 2050
size, 2050
swap, 2050
std::destroying_delete_t, 2051
std::discard_block_engine< _RandomNumberEngine, __p, __r >, 2056
base, 2058
discard, 2058
discard_block_engine, 2057, 2058
max, 2059
min, 2059
operator<<, 2060
operator>>, 2060
operator(), 2059
operator==, 2060
result_type, 2057
seed, 2059
std::discrete_distribution< _IntType >, 2061
max, 2062
min, 2062
operator<<, 2063
operator>>, 2064
operator(), 2062
operator==, 2063
param, 2062, 2063
probabilities, 2063
reset, 2063
result_type, 2062
std::discrete_distribution< _IntType >::param_type, 2722
std::divides< _Tp >, 2064
first_argument_type, 2065
result_type, 2065
second_argument_type, 2065
std::divides< void >, 2065
first_argument_type, 2066
result_type, 2066
second_argument_type, 2066
std::domain_error, 2066
what, 2067
std::enable_if< bool, _Tp >, 2083
std::enable_shared_from_this< _Tp >, 2083
std::encoding_state, 2104
std::equal_to< _Tp >, 2109
first_argument_type, 2109
result_type, 2109
second_argument_type, 2109
std::error_category, 2110
default_error_condition, 2111
equivalent, 2111
message, 2111
name, 2111
operator<=>, 2111
operator==, 2111
std::error_code, 2111
category, 2112
default_error_condition, 2112
error_code, 2112

message, 2113
operator bool, 2113
value, 2113
std::error_condition, 2113
assign, 2114
category, 2114
clear, 2114
error_condition, 2114
message, 2114
operator bool, 2114
value, 2115
std::exception, 2116
what, 2117
std::exception_ptr, 2118
std::experimental, 691
gcd, 701
get_deleter, 701
is_bind_expression_v, 703
is_placeholder_v, 703
lcm, 701
make_boyer_moore_horspool_searcher, 701
make_boyer_moore_searcher, 702
make_default_searcher, 702
make_ostream_joiner, 702
not_fn, 702
sample, 702
std::experimental::filesystem::v1::basic_string_view< _CharT, _Traits >, 1646
std::experimental::filesystem::v1::filesystem_error, 2128
what, 2129
std::experimental::filesystem::v1::path, 2736
std::experimental::filesystem::v1::path::iterator, 2298
std::experimental::filesystem::v1::space_info, 2979
std::experimental::fundamentals_v1::any, 1064
~any, 1065
any, 1065
clear, 1065
empty, 1065
operator=, 1065, 1066
swap, 1066
type, 1066
std::experimental::fundamentals_v1::bad_any_cast, 1079
what, 1080
std::experimental::fundamentals_v1::bad_optional_access, 1083
what, 1084
std::experimental::fundamentals_v1::basic_string_view< _CharT, _Traits >, 1648
std::experimental::fundamentals_v1::in_place_t, 2241
std::experimental::fundamentals_v1::nullopt_t, 2544
std::experimental::fundamentals_v1::optional< _Tp >, 2685
std::experimental::fundamentals_v2::ostream_joiner< _DelimT, _CharT, _Traits >, 2689
std::experimental::fundamentals_v2::owner_less< shared_ptr< _Tp > >, 2704
first_argument_type, 2704
result_type, 2704
second_argument_type, 2704
std::experimental::fundamentals_v2::owner_less< weak_ptr< _Tp > >, 2706
first_argument_type, 2706
result_type, 2706
second_argument_type, 2706
std::experimental::fundamentals_v2::propagate_const< _Tp >, 2774
std::exponential_distribution< _RealType >, 2118
exponential_distribution, 2120
lambda, 2120
max, 2120
min, 2120
operator(), 2120
operator==, 2121
param, 2120, 2121
reset, 2121
result_type, 2120
std::exponential_distribution< _RealType >::param_type, 2723
std::extent< typename, _Uint >, 2121
std::extreme_value_distribution< _RealType >, 2122
a, 2123
b, 2123
max, 2124
min, 2124
operator(), 2124
operator==, 2125
param, 2124
reset, 2124
result_type, 2123
std::extreme_value_distribution< _RealType >::param_type, 2723
std::filesystem, 703
std::filesystem::directory_entry, 2054
std::filesystem::directory_iterator, 2055
std::filesystem::file_status, 2128
std::filesystem::filesystem_error, 2130
what, 2130
std::filesystem::path, 2739
format, 2741
operator<<, 2742
operator<=>, 2742
operator>>, 2742
operator/, 2742
operator==, 2742
std::filesystem::path::iterator, 2298
std::filesystem::recursive_directory_iterator, 2810
std::filesystem::space_info, 2979
std::fisher_f_distribution< _RealType >, 2131

max, 2133
min, 2133
operator<<, 2133
operator>>, 2134
operator(), 2133
operator==, 2134
param, 2133
reset, 2133
result_type, 2132
std::fisher_f_distribution< _RealType >::param_type, 2724
std::forward_iterator_tag, 2135
std::forward_list< _Tp, _Alloc >, 2139
~forward_list, 2145
assign, 2145, 2146
before_begin, 2146
begin, 2147
cbefore_begin, 2147
cbegin, 2147
cend, 2147
clear, 2147
emplace_after, 2147
emplace_front, 2148
empty, 2148
end, 2148
erase_after, 2149
forward_list, 2142, 2143, 2145
front, 2149, 2150
get_allocator, 2150
insert_after, 2150, 2151
max_size, 2151
merge, 2152
operator=, 2152, 2153
pop_front, 2153
push_front, 2153
remove, 2153
remove_if, 2154
resize, 2154
reverse, 2155
sort, 2155
splice_after, 2155, 2156
swap, 2156
unique, 2157
std::fpos< _StateT >, 2157
fpos, 2158
operator streamoff, 2158
operator+, 2158
operator+=, 2158
operator-, 2158, 2159
operator-=, 2159
state, 2159
std::from_chars_result, 2161
std::front_insert_iterator< _Container >, 2161
container_type, 2162
front_insert_iterator, 2163
iterator_category, 2162
operator++, 2163
operator=, 2163
operator*, 2163
pointer, 2162
reference, 2162
value_type, 2162
std::function< _Res(_ArgTypes...)>, 2164
function, 2165, 2166
operator bool, 2166
operator(), 2166
operator=, 2167, 2168
swap, 2168
target, 2169
target_type, 2169
std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>, 2170
function_ref, 2171
operator(), 2172
std::future< _Res >, 2172
_M_get_result, 2173
future, 2173
get, 2173
std::future< _Res & >, 2173
_M_get_result, 2175
future, 2175
get, 2175
std::future< void >, 2175
_M_get_result, 2177
future, 2177
get, 2177
std::future_error, 2177
what, 2178
std::gamma_distribution< _RealType >, 2178
alpha, 2180
beta, 2180
gamma_distribution, 2180
max, 2180
min, 2180
operator<<, 2181
operator>>, 2182
operator(), 2180, 2181
operator==, 2182
param, 2181
reset, 2181
result_type, 2180
std::gamma_distribution< _RealType >::param_type, 2724
std::geometric_distribution< _IntType >, 2182
max, 2183
min, 2183
operator(), 2184
operator==, 2184

p, 2184
param, 2184
reset, 2184
result_type, 2183
std::geometric_distribution< _IntType >::param_type, 2725
std::greater< _Tp >, 2197
first_argument_type, 2197
result_type, 2197
second_argument_type, 2197
std::greater< void >, 2198
first_argument_type, 2199
result_type, 2199
second_argument_type, 2199
std::greater_equal< _Tp >, 2199
first_argument_type, 2200
result_type, 2200
second_argument_type, 2200
std::greater_equal< void >, 2201
first_argument_type, 2201
result_type, 2201
second_argument_type, 2201
std::gslice, 2203
std::gslice_array< _Tp >, 2203
operator<<=, 2205
operator>>=, 2205
operator+=, 2205
operator-=, 2205
operator/=, 2205
operator%=, 2204
operator&=, 2204
operator*=, 2205
operator^=, 2205
operator|=, 2205
std::has_virtual_destructor< _Tp >, 2206
std::hash< __debug::bitset< _Nb > >, 2206
std::hash< __debug::vector< bool, _Alloc > >, 2207
std::hash< __gnu_cxx::__u16vstring >, 2207
std::hash< __gnu_cxx::__u32vstring >, 2207
std::hash< __gnu_cxx::__vstring >, 2207
std::hash< __gnu_cxx::__wvstring >, 2208
std::hash< __gnu_cxx::throw_value_limit >, 2208
argument_type, 2209
result_type, 2209
std::hash< __gnu_cxx::throw_value_random >, 2209
argument_type, 2210
result_type, 2210
std::hash< __gnu_debug::basic_string< _CharT > >, 2210
std::hash< __shared_ptr< _Tp, _Lp > >, 2210
std::hash< _Tp >, 2206
std::hash< _Tp * >, 2211
std::hash< basic_string< char, char_traits< char >, _Alloc > >, 2211
std::hash< basic_string< char16_t, char_traits< char16_t >, _Alloc > >, 2211
std::hash< basic_string< char32_t, char_traits< char32_t >, _Alloc > >, 2212
std::hash< basic_string< wchar_t, char_traits< wchar_t >, _Alloc > >, 2212
std::hash< bool >, 2212
std::hash< char >, 2213
std::hash< char16_t >, 2213
std::hash< char32_t >, 2213
std::hash< double >, 2213
std::hash< error_code >, 2214
std::hash< error_condition >, 2214
std::hash< experimental::optional< _Tp > >, 2214
std::hash< experimental::shared_ptr< _Tp > >, 2215
std::hash< float >, 2215
std::hash< int >, 2215
std::hash< long >, 2216
std::hash< long double >, 2216
std::hash< long long >, 2216
std::hash< shared_ptr< _Tp > >, 2216
std::hash< short >, 2217
std::hash< signed char >, 2217
std::hash< thread::id >, 2217
std::hash< type_index >, 2217
std::hash< unique_ptr< _Tp, _Dp > >, 2218
std::hash< unsigned char >, 2218
std::hash< unsigned int >, 2218
std::hash< unsigned long >, 2219
std::hash< unsigned long long >, 2219
std::hash< unsigned short >, 2219
std::hash< wchar_t >, 2219
std::hash<::bitset< _Nb > >, 2220
std::hash<::vector< bool, _Alloc > >, 2220
std::identity, 2241
std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >, 2241
base, 2243
discard, 2243
independent_bits_engine, 2242, 2243
max, 2243
min, 2244
operator>>, 2245
operator(), 2244
operator==, 2244
result_type, 2242
seed, 2244
std::indirect_array< _Tp >, 2245
operator<<=, 2247
operator>>=, 2247
operator+=, 2247
operator-=, 2247
operator/=, 2247
operator%=, 2247

operator&=, 2247
operator*=, 2247
operator^=, 2248
operator|=, 2248
std::initializer_list< _E >, 2248
begin, 2249
end, 2249
std::input_iterator_tag, 2249
std::insert_iterator< _Container >, 2251
container_type, 2253
insert_iterator, 2253
iterator_category, 2253
operator++, 2253
operator=, 2254
operator*, 2253
pointer, 2253
reference, 2253
value_type, 2253
std::integer_sequence< _Tp, _Idx >, 2254
std::integral_constant< _Tp, __v >, 2255
std::invalid_argument, 2255
what, 2256
std::ios_base, 2256
_M_getloc, 2261
~ios_base, 2261
adjustfield, 2266
app, 2266
ate, 2266
badbit, 2266
basefield, 2266
beg, 2266
binary, 2266
boolalpha, 2267
cur, 2267
dec, 2267
end, 2267
eofbit, 2267
event, 2260
event_callback, 2259
failbit, 2267
fixed, 2268
flags, 2261
floatfield, 2268
getloc, 2261
goodbit, 2268
hex, 2268
imbue, 2262
in, 2268
internal, 2268
iostate, 2260
iword, 2262
left, 2269
oct, 2269
openmode, 2260
out, 2269
precision, 2262, 2263
pword, 2263
register_callback, 2263
right, 2269
scientific, 2269
seekdir, 2260
setf, 2264
showbase, 2269
showpoint, 2269
showpos, 2269
skipws, 2269
sync_with_stdio, 2264
trunc, 2270
unitbuf, 2270
unsetf, 2265
uppercase, 2270
width, 2265
xalloc, 2265
std::is_abstract< _Tp >, 2270
std::is_arithmetic< _Tp >, 2270
std::is_array< _Tp >, 2270
std::is_assignable< _Tp, _Up >, 2271
std::is_base_of< _Base, _Derived >, 2271
std::is_bind_expression< _Bind< _Signature > >, 2272
std::is_bind_expression< _Bind_result< _Result, _Signature > >, 2272
std::is_bind_expression< _Tp >, 2271
std::is_bind_expression< const _Bind< _Signature > >, 2272
std::is_bind_expression< const _Bind_result< _Result, _Signature > >, 2272
std::is_bind_expression< const volatile _Bind< _Signature > >, 2273
std::is_bind_expression< const volatile _Bind_result< _Result, _Signature > >, 2273
std::is_bind_expression< volatile _Bind< _Signature > >, 2273
std::is_bind_expression< volatile _Bind_result< _Result, _Signature > >, 2273
std::is_class< _Tp >, 2274
std::is_compound< _Tp >, 2274
std::is_const< _Tp >, 2274
std::is_constructible< _Tp, _Args >, 2274
std::is_copy_assignable< _Tp >, 2274
std::is_copy_constructible< _Tp >, 2275
std::is_default_constructible< _Tp >, 2275
std::is_destructible< _Tp >, 2275
std::is_empty< _Tp >, 2275
std::is_enum< _Tp >, 2276
std::is_error_code_enum< _Tp >, 2276
std::is_error_code_enum< future_errc >, 2276
std::is_error_condition_enum< _Tp >, 2276
std::is_floating_point< _Tp >, 2277

std::is_function< _Tp >, 2277
std::is_fundamental< _Tp >, 2278
std::is_integral< _Tp >, 2278
std::is_layout_compatible< _Tp, _Up >, 2278
std::is_literal_type< _Tp >, 2278
std::is_lvalue_reference< typename >, 2279
std::is_member_function_pointer< _Tp >, 2279
std::is_member_object_pointer< _Tp >, 2279
std::is_member_pointer< _Tp >, 2279
std::is_move_assignable< _Tp >, 2280
std::is_move_constructible< _Tp >, 2280
std::is_nothrow_assignable< _Tp, _Up >, 2280
std::is_nothrow_constructible< _Tp, _Args >, 2280
std::is_nothrow_copy_assignable< _Tp >, 2280
std::is_nothrow_copy_constructible< _Tp >, 2281
std::is_nothrow_default_constructible< _Tp >, 2281
std::is_nothrow_destructible< _Tp >, 2281
std::is_nothrow_move_assignable< _Tp >, 2281
std::is_nothrow_move_constructible< _Tp >, 2282
std::is_object< _Tp >, 2282
std::is_placeholder< _Placeholder< _Num > >, 2283
std::is_placeholder< _Tp >, 2282
std::is_pod< _Tp >, 2284
std::is_pointer< _Tp >, 2285
std::is_pointer_interconvertible_base_of< _Base, _Derived >, 2285
std::is_polymorphic< _Tp >, 2285
std::is_reference< _Tp >, 2285
std::is_rvalue_reference< typename >, 2286
std::is_same< _Tp, _Up >, 2286
std::is_scalar< _Tp >, 2286
std::is_signed< _Tp >, 2286
std::is_standard_layout< _Tp >, 2287
std::is_trivial< _Tp >, 2287
std::is_trivially_assignable< _Tp, _Up >, 2287
std::is_trivially_constructible< _Tp, _Args >, 2287
std::is_trivially_copy_assignable< _Tp >, 2288
std::is_trivially_copy_constructible< _Tp >, 2288
std::is_trivially_copyable< _Tp >, 2288
std::is_trivially_default_constructible< _Tp >, 2288
std::is_trivially_destructible< _Tp >, 2289
std::is_trivially_move_assignable< _Tp >, 2289
std::is_trivially_move_constructible< _Tp >, 2289
std::is_union< _Tp >, 2289
std::is_unsigned< _Tp >, 2290
std::is_void< _Tp >, 2290
std::is_volatile< _Tp >, 2290
std::istream_iterator< _Tp, _CharT, _Traits, _Dist >, 2290
  difference_type, 2292
  istream_iterator, 2292
  iterator_category, 2292
  operator==, 2292
  pointer, 2292
  reference, 2292
  value_type, 2292
std::istreambuf_iterator< _CharT, _Traits >, 2293
  char_type, 2294
  difference_type, 2294
  equal, 2296
  int_type, 2294
  istream_type, 2294
  istreambuf_iterator, 2295, 2296
  iterator_category, 2295
  operator++, 2296
  operator*, 2296
  pointer, 2295
  reference, 2295
  streambuf_type, 2295
  traits_type, 2295
  value_type, 2295
std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >, 2299
  difference_type, 2300
  iterator_category, 2300
  pointer, 2300
  reference, 2300
  value_type, 2300
std::iterator_traits< _Iterator >, 2301
std::iterator_traits< _Tp * >, 2301
std::length_error, 2311
  what, 2312
std::less< _Tp >, 2312
  first_argument_type, 2313
  result_type, 2313
  second_argument_type, 2313
std::less_equal< _Tp >, 2314
  first_argument_type, 2315
  result_type, 2315
  second_argument_type, 2315
std::less_equal< void >, 2315
  first_argument_type, 2316
  result_type, 2316
  second_argument_type, 2316
std::linear_congruential_engine< _UIntType, __a, __c, __m >, 2317
  discard, 2319
  increment, 2323
  linear_congruential_engine, 2319
  max, 2319
  min, 2320
  modulus, 2323
  multiplier, 2323
  operator<<, 2321
  operator>>, 2321
  operator(), 2320
  operator==, 2321
  result_type, 2319
  seed, 2320

std::list< _Tp, _Alloc >, 2326
_M_create_node, 2332
~list, 2332
assign, 2334
back, 2335
begin, 2335
cbegin, 2335
cend, 2335
clear, 2335
crbegin, 2335
crend, 2336
emplace, 2336
empty, 2336
end, 2336
erase, 2336, 2337
front, 2337
get_allocator, 2337
insert, 2338, 2339
list, 2330–2332
max_size, 2340
merge, 2340
operator=, 2341
pop_back, 2341
pop_front, 2342
push_back, 2342
push_front, 2342
rbegin, 2342, 2343
remove, 2343
remove_if, 2343
rend, 2343
resize, 2344
reverse, 2344
size, 2344
sort, 2344, 2345
splice, 2345, 2346
swap, 2346
unique, 2347
std::literals, 706
std::literals::chrono_literals, 707
std::locale, 2349
~locale, 2353
all, 2357
category, 2351
classic, 2353
collate, 2357
combine, 2353
ctype, 2357
global, 2354
has_facet, 2356
locale, 2351–2353
messages, 2357
monetary, 2357
name, 2354
none, 2357
numeric, 2357
operator(), 2354
operator=, 2355
operator==, 2355
time, 2358
use_facet, 2356
std::locale::facet, 2125
~facet, 2127
facet, 2127
std::locale::id, 2238
has_facet, 2239
id, 2239
use_facet, 2239
std::lock_guard< _Mutex >, 2358
std::logic_error, 2358
logic_error, 2359
what, 2359
std::logical_and< _Tp >, 2360
first_argument_type, 2360
result_type, 2360
second_argument_type, 2360
std::logical_and< void >, 2361
first_argument_type, 2361
result_type, 2361
second_argument_type, 2362
std::logical_not< _Tp >, 2362
argument_type, 2362
result_type, 2362
std::logical_not< void >, 2363
argument_type, 2363
result_type, 2363
std::logical_or< _Tp >, 2364
first_argument_type, 2364
result_type, 2364
second_argument_type, 2364
std::logical_or< void >, 2365
first_argument_type, 2365
result_type, 2365
second_argument_type, 2366
std::lognormal_distribution< _RealType >, 2366
max, 2367
min, 2367
operator<<, 2368
operator>>, 2368
operator(), 2367
operator==, 2368
param, 2367
reset, 2368
result_type, 2367
std::lognormal_distribution< _RealType >::param_type, 2725
std::make_signed< _Tp >, 2376
std::make_unsigned< _Tp >, 2376
std::map< _Key, _Tp, _Compare, _Alloc >, 2380

__attribute, 2386
~map, 2386
at, 2387
begin, 2387
cbegin, 2387
cend, 2387
clear, 2388
contains, 2388
count, 2388, 2389
crbegin, 2389
crend, 2389
emplace, 2389
emplace_hint, 2390
empty, 2390
end, 2391
equal_range, 2391, 2392
erase, 2393
find, 2394, 2395
get_allocator, 2395
insert, 2395–2398
insert_or_assign, 2399
key_comp, 2400
lower_bound, 2400, 2401
map, 2383–2386
max_size, 2401
operator=, 2401, 2402
operator[], 2402
rbegin, 2402, 2403
rend, 2403
size, 2403
swap, 2403
upper_bound, 2403, 2404
value_comp, 2405
std::mask_array< _Tp >, 2405
operator<<=, 2407
operator>>=, 2407
operator+=, 2407
operator-=, 2407
operator/=, 2407
operator%=, 2406
operator&=, 2406
operator*=, 2407
operator^=, 2407
operator|=, 2407
std::match_results< _Bi_iter, _Alloc >, 2409
~match_results, 2412
begin, 2412
cbegin, 2412
cend, 2413
empty, 2413
end, 2413
format, 2413, 2414
get_allocator, 2414
length, 2414
match_results, 2411, 2412
max_size, 2414
operator=, 2415
operator[], 2415
position, 2415
prefix, 2416
ready, 2416
size, 2416
str, 2416
suffix, 2417
swap, 2417
std::mem_fun1_ref_t< _Ret, _Tp, _Arg >, 2419
first_argument_type, 2420
result_type, 2420
second_argument_type, 2420
std::mem_fun1_t< _Ret, _Tp, _Arg >, 2421
first_argument_type, 2421
result_type, 2421
second_argument_type, 2421
std::mem_fun_ref_t< _Ret, _Tp >, 2422
argument_type, 2422
result_type, 2422
std::mem_fun_t< _Ret, _Tp >, 2423
argument_type, 2423
result_type, 2423
std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >, 2424
discard, 2427
max, 2427
mersenne_twister_engine, 2427
min, 2427
operator<<, 2428
operator>>, 2428
operator==, 2428
result_type, 2426
std::messages< _CharT >, 2429
~messages, 2431
char_type, 2431
do_get, 2432
id, 2432
messages, 2431
string_type, 2431
std::messages_base, 2432
std::messages_byname< _CharT >, 2433
do_get, 2434
id, 2434
std::minus< _Tp >, 2435
first_argument_type, 2435
result_type, 2435
second_argument_type, 2435
std::minus< void >, 2436
first_argument_type, 2436
result_type, 2436

second_argument_type, 2436
std::modulus< _Tp >, 2437
first_argument_type, 2438
result_type, 2438
second_argument_type, 2438
std::modulus< void >, 2438
first_argument_type, 2439
result_type, 2439
second_argument_type, 2439
std::money_base, 2439
std::money_get< _CharT, _InIter >, 2441
~money_get, 2442
char_type, 2442
do_get, 2443
get, 2443, 2444
id, 2445
iter_type, 2442
money_get, 2442
string_type, 2442
std::money_put< _CharT, _OutIter >, 2445
~money_put, 2447
char_type, 2446
do_put, 2447
id, 2449
iter_type, 2446
money_put, 2447
put, 2448, 2449
string_type, 2446
std::moneypunct< _CharT, _Intl >, 2449
~moneypunct, 2452
char_type, 2451
curr_symbol, 2453
decimal_point, 2453
do_curr_symbol, 2453
do_decimal_point, 2453
do_frac_digits, 2454
do_grouping, 2454
do_neg_format, 2454
do_negative_sign, 2454
do_pos_format, 2455
do_positive_sign, 2455
do_thousands_sep, 2455
frac_digits, 2456
grouping, 2456
id, 2458
intl, 2458
moneypunct, 2452
neg_format, 2456
negative_sign, 2457
pos_format, 2457
positive_sign, 2457
string_type, 2451
thousands_sep, 2458
std::moneypunct_byname< _CharT, _Intl >, 2458
curr_symbol, 2460
decimal_point, 2460
do_curr_symbol, 2461
do_decimal_point, 2461
do_frac_digits, 2461
do_grouping, 2462
do_neg_format, 2462
do_negative_sign, 2462
do_pos_format, 2462
do_positive_sign, 2463
do_thousands_sep, 2463
frac_digits, 2463
grouping, 2464
id, 2466
neg_format, 2464
negative_sign, 2464
pos_format, 2465
positive_sign, 2465
thousands_sep, 2465
std::move_iterator< _Iterator >, 2467
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>, 2469
move_only_function, 2470, 2471
operator bool, 2471
operator(), 2471
operator=, 2471, 2472
operator==, 2472
swap, 2472
std::move_sentinel< _Sent >, 2473
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2476
~multimap, 2482
begin, 2482
cbegin, 2482
cend, 2482
clear, 2482
contains, 2483
count, 2483, 2484
crbegin, 2484
crend, 2484
emplace, 2484
emplace_hint, 2485
empty, 2485
end, 2485
equal_range, 2486, 2487
erase, 2487, 2488
find, 2489, 2490
get_allocator, 2490
insert, 2490–2493
key_comp, 2494
lower_bound, 2494, 2495
max_size, 2495
multimap, 2479–2481
operator=, 2496
rbegin, 2496

rend, 2497
size, 2497
swap, 2497
upper_bound, 2497, 2498
value_comp, 2499
std::multiplies< _Tp >, 2499
first_argument_type, 2500
result_type, 2500
second_argument_type, 2500
std::multiplies< void >, 2500
first_argument_type, 2501
result_type, 2501
second_argument_type, 2501
std::multiset< _Key, _Compare, _Alloc >, 2504
~multiset, 2509
begin, 2510
cbegin, 2510
cend, 2510
clear, 2510
contains, 2510, 2511
count, 2511
crbegin, 2511
crend, 2512
emplace, 2512
emplace_hint, 2512
empty, 2513
end, 2513
equal_range, 2513, 2514
erase, 2515
find, 2516, 2517
get_allocator, 2517
insert, 2517–2519
key_comp, 2519
lower_bound, 2519, 2520
max_size, 2521
multiset, 2507–2509
operator=, 2521
rbegin, 2521
rend, 2522
size, 2522
swap, 2522
upper_bound, 2522, 2523
value_comp, 2523
std::mutex, 2527
std::negate< _Tp >, 2527
argument_type, 2528
result_type, 2528
std::negate< void >, 2528
argument_type, 2529
result_type, 2529
std::negative_binomial_distribution< _IntType >, 2529
k, 2531
max, 2531
min, 2531
operator<<, 2532
operator>>, 2532
operator(), 2531
operator==, 2532
p, 2531
param, 2531
reset, 2532
result_type, 2531
std::negative_binomial_distribution< _IntType >::param_type, 2726
std::nested_exception, 2533
nested_exception, 2533
nested_ptr, 2533
rethrow_nested, 2533
std::normal_distribution< _RealType >, 2536
max, 2537
mean, 2537
min, 2538
normal_distribution, 2537
operator<<, 2539
operator>>, 2539
operator(), 2538
operator==, 2539
param, 2538
reset, 2538
result_type, 2537
stddev, 2538
std::normal_distribution< _RealType >::param_type, 2726
std::not_equal_to< _Tp >, 2540
first_argument_type, 2540
result_type, 2540
second_argument_type, 2541
std::not_equal_to< void >, 2541
first_argument_type, 2542
result_type, 2542
second_argument_type, 2542
std::num_get< _CharT, _InIter >, 2545
~num_get, 2547
char_type, 2547
do_get, 2547–2553
get, 2554–2560
id, 2560
iter_type, 2547
num_get, 2547
std::num_put< _CharT, _OutIter >, 2560
~num_put, 2563
char_type, 2562
do_put, 2563–2567
id, 2573
iter_type, 2562
num_put, 2563
put, 2567–2572
std::numeric_limits< _Tp >, 2573

denorm_min, 2574
digits, 2575
digits10, 2576
epsilon, 2574
has_denorm, 2576
has_denorm_loss, 2576
has_infinity, 2576
has_quiet_NaN, 2576
has_signaling_NaN, 2576
infinity, 2574
is_bounded, 2576
is_exact, 2576
is_iec559, 2576
is_integer, 2576
is_modulo, 2577
is_signed, 2577
is_specialized, 2577
lowest, 2574
max, 2575
max_digits10, 2577
max_exponent, 2577
max_exponent10, 2577
min, 2575
min_exponent, 2577
min_exponent10, 2577
quiet_NaN, 2575
radix, 2577
round_error, 2575
round_style, 2577
signaling_NaN, 2575
tinyness_before, 2578
traps, 2578
std::numeric_limits< bool >, 2578
denorm_min, 2580
digits, 2581
digits10, 2581
epsilon, 2580
has_denorm, 2581
has_denorm_loss, 2581
has_infinity, 2581
has_quiet_NaN, 2581
has_signaling_NaN, 2581
infinity, 2580
is_bounded, 2581
is_exact, 2581
is_iec559, 2582
is_integer, 2582
is_modulo, 2582
is_signed, 2582
is_specialized, 2582
lowest, 2580
max, 2580
max_digits10, 2582
max_exponent, 2582
max_exponent10, 2582
min, 2580
min_exponent, 2582
min_exponent10, 2583
quiet_NaN, 2580
radix, 2583
round_error, 2580
round_style, 2583
signaling_NaN, 2581
tinyness_before, 2583
traps, 2583
std::numeric_limits< char >, 2583
denorm_min, 2585
digits, 2586
digits10, 2586
epsilon, 2585
has_denorm, 2586
has_denorm_loss, 2586
has_infinity, 2586
has_quiet_NaN, 2586
has_signaling_NaN, 2586
infinity, 2585
is_bounded, 2587
is_exact, 2587
is_iec559, 2587
is_integer, 2587
is_modulo, 2587
is_signed, 2587
is_specialized, 2587
lowest, 2585
max, 2585
max_digits10, 2587
max_exponent, 2587
max_exponent10, 2587
min, 2585
min_exponent, 2588
min_exponent10, 2588
quiet_NaN, 2586
radix, 2588
round_error, 2586
round_style, 2588
signaling_NaN, 2586
tinyness_before, 2588
traps, 2588
std::numeric_limits< char16_t >, 2588
denorm_min, 2590
digits, 2591
digits10, 2591
epsilon, 2590
has_denorm, 2591
has_denorm_loss, 2592
has_infinity, 2592
has_quiet_NaN, 2592
has_signaling_NaN, 2592

infinity, 2590
is_bounded, 2592
is_exact, 2592
is_iec559, 2592
is_integer, 2592
is_modulo, 2592
is_signed, 2592
is_specialized, 2593
lowest, 2591
max, 2591
max_digits10, 2593
max_exponent, 2593
max_exponent10, 2593
min, 2591
min_exponent, 2593
min_exponent10, 2593
quiet_NaN, 2591
radix, 2593
round_error, 2591
round_style, 2593
signaling_NaN, 2591
tinyness_before, 2593
traps, 2593
std::numeric_limits< char32_t >, 2594
denorm_min, 2596
digits, 2597
digits10, 2597
epsilon, 2596
has_denorm, 2597
has_denorm_loss, 2597
has_infinity, 2597
has_quiet_NaN, 2597
has_signaling_NaN, 2597
infinity, 2596
is_bounded, 2597
is_exact, 2597
is_iec559, 2597
is_integer, 2597
is_modulo, 2598
is_signed, 2598
is_specialized, 2598
lowest, 2596
max, 2596
max_digits10, 2598
max_exponent, 2598
max_exponent10, 2598
min, 2596
min_exponent, 2598
min_exponent10, 2598
quiet_NaN, 2596
radix, 2598
round_error, 2596
round_style, 2598
signaling_NaN, 2596
tinyness_before, 2599
traps, 2599
std::numeric_limits< double >, 2599
denorm_min, 2601
digits, 2602
digits10, 2602
epsilon, 2601
has_denorm, 2602
has_denorm_loss, 2602
has_infinity, 2602
has_quiet_NaN, 2602
has_signaling_NaN, 2602
infinity, 2601
is_bounded, 2602
is_exact, 2602
is_iec559, 2603
is_integer, 2603
is_modulo, 2603
is_signed, 2603
is_specialized, 2603
lowest, 2601
max, 2601
max_digits10, 2603
max_exponent, 2603
max_exponent10, 2603
min, 2601
min_exponent, 2603
min_exponent10, 2604
quiet_NaN, 2601
radix, 2604
round_error, 2601
round_style, 2604
signaling_NaN, 2602
tinyness_before, 2604
traps, 2604
std::numeric_limits< float >, 2604
denorm_min, 2606
digits, 2607
digits10, 2607
epsilon, 2606
has_denorm, 2607
has_denorm_loss, 2607
has_infinity, 2607
has_quiet_NaN, 2607
has_signaling_NaN, 2607
infinity, 2606
is_bounded, 2608
is_exact, 2608
is_iec559, 2608
is_integer, 2608
is_modulo, 2608
is_signed, 2608
is_specialized, 2608
lowest, 2606

max, 2606
max_digits10, 2608
max_exponent, 2608
max_exponent10, 2608
min, 2606
min_exponent, 2609
min_exponent10, 2609
quiet_NaN, 2607
radix, 2609
round_error, 2607
round_style, 2609
signaling_NaN, 2607
tinyness_before, 2609
traps, 2609
std::numeric_limits< int >, 2609
denorm_min, 2611
digits, 2612
digits10, 2612
epsilon, 2611
has_denorm, 2612
has_denorm_loss, 2612
has_infinity, 2613
has_quiet_NaN, 2613
has_signaling_NaN, 2613
infinity, 2611
is_bounded, 2613
is_exact, 2613
is_iec559, 2613
is_integer, 2613
is_modulo, 2613
is_signed, 2613
is_specialized, 2613
lowest, 2612
max, 2612
max_digits10, 2614
max_exponent, 2614
max_exponent10, 2614
min, 2612
min_exponent, 2614
min_exponent10, 2614
quiet_NaN, 2612
radix, 2614
round_error, 2612
round_style, 2614
signaling_NaN, 2612
tinyness_before, 2614
traps, 2614
std::numeric_limits< long >, 2615
denorm_min, 2616
digits, 2617
digits10, 2617
epsilon, 2616
has_denorm, 2618
has_denorm_loss, 2618
has_infinity, 2618
has_quiet_NaN, 2618
has_signaling_NaN, 2618
infinity, 2617
is_bounded, 2618
is_exact, 2618
is_iec559, 2618
is_integer, 2618
is_modulo, 2618
is_signed, 2619
is_specialized, 2619
lowest, 2617
max, 2617
max_digits10, 2619
max_exponent, 2619
max_exponent10, 2619
min, 2617
min_exponent, 2619
min_exponent10, 2619
quiet_NaN, 2617
radix, 2619
round_error, 2617
round_style, 2619
signaling_NaN, 2617
tinyness_before, 2620
traps, 2620
std::numeric_limits< long double >, 2620
denorm_min, 2622
digits, 2623
digits10, 2623
epsilon, 2622
has_denorm, 2623
has_denorm_loss, 2623
has_infinity, 2623
has_quiet_NaN, 2623
has_signaling_NaN, 2623
infinity, 2622
is_bounded, 2623
is_exact, 2623
is_iec559, 2624
is_integer, 2624
is_modulo, 2624
is_signed, 2624
is_specialized, 2624
lowest, 2622
max, 2622
max_digits10, 2624
max_exponent, 2624
max_exponent10, 2624
min, 2622
min_exponent, 2624
min_exponent10, 2625
quiet_NaN, 2622
radix, 2625

round_error, 2622
round_style, 2625
signaling_NaN, 2623
tinyness_before, 2625
traps, 2625
std::numeric_limits< long long >, 2625
denorm_min, 2627
digits, 2628
digits10, 2628
epsilon, 2627
has_denorm, 2628
has_denorm_loss, 2628
has_infinity, 2628
has_quiet_NaN, 2628
has_signaling_NaN, 2628
infinity, 2627
is_bounded, 2629
is_exact, 2629
is_iec559, 2629
is_integer, 2629
is_modulo, 2629
is_signed, 2629
is_specialized, 2629
lowest, 2627
max, 2627
max_digits10, 2629
max_exponent, 2629
max_exponent10, 2629
min, 2627
min_exponent, 2630
min_exponent10, 2630
quiet_NaN, 2628
radix, 2630
round_error, 2628
round_style, 2630
signaling_NaN, 2628
tinyness_before, 2630
traps, 2630
std::numeric_limits< short >, 2630
denorm_min, 2632
digits, 2633
digits10, 2633
epsilon, 2632
has_denorm, 2633
has_denorm_loss, 2634
has_infinity, 2634
has_quiet_NaN, 2634
has_signaling_NaN, 2634
infinity, 2632
is_bounded, 2634
is_exact, 2634
is_iec559, 2634
is_integer, 2634
is_modulo, 2634
is_signed, 2634
is_specialized, 2635
lowest, 2633
max, 2633
max_digits10, 2635
max_exponent, 2635
max_exponent10, 2635
min, 2633
min_exponent, 2635
min_exponent10, 2635
quiet_NaN, 2633
radix, 2635
round_error, 2633
round_style, 2635
signaling_NaN, 2633
tinyness_before, 2635
traps, 2635
std::numeric_limits< signed char >, 2636
denorm_min, 2638
digits, 2639
digits10, 2639
epsilon, 2638
has_denorm, 2639
has_denorm_loss, 2639
has_infinity, 2639
has_quiet_NaN, 2639
has_signaling_NaN, 2639
infinity, 2638
is_bounded, 2639
is_exact, 2639
is_iec559, 2639
is_integer, 2639
is_modulo, 2640
is_signed, 2640
is_specialized, 2640
lowest, 2638
max, 2638
max_digits10, 2640
max_exponent, 2640
max_exponent10, 2640
min, 2638
min_exponent, 2640
min_exponent10, 2640
quiet_NaN, 2638
radix, 2640
round_error, 2638
round_style, 2640
signaling_NaN, 2638
tinyness_before, 2641
traps, 2641
std::numeric_limits< unsigned char >, 2641
denorm_min, 2643
digits, 2644
digits10, 2644

epsilon, 2643
has_denorm, 2644
has_denorm_loss, 2644
has_infinity, 2644
has_quiet_NaN, 2644
has_signaling_NaN, 2644
infinity, 2643
is_bounded, 2644
is_exact, 2644
is_iec559, 2645
is_integer, 2645
is_modulo, 2645
is_signed, 2645
is_specialized, 2645
lowest, 2643
max, 2643
max_digits10, 2645
max_exponent, 2645
max_exponent10, 2645
min, 2643
min_exponent, 2645
min_exponent10, 2646
quiet_NaN, 2643
radix, 2646
round_error, 2643
round_style, 2646
signaling_NaN, 2644
tinyness_before, 2646
traps, 2646
std::numeric_limits< unsigned int >, 2646
denorm_min, 2648
digits, 2649
digits10, 2649
epsilon, 2648
has_denorm, 2649
has_denorm_loss, 2649
has_infinity, 2649
has_quiet_NaN, 2649
has_signaling_NaN, 2649
infinity, 2648
is_bounded, 2650
is_exact, 2650
is_iec559, 2650
is_integer, 2650
is_modulo, 2650
is_signed, 2650
is_specialized, 2650
lowest, 2648
max, 2648
max_digits10, 2650
max_exponent, 2650
max_exponent10, 2650
min, 2648
min_exponent, 2651
min_exponent10, 2651
quiet_NaN, 2649
radix, 2651
round_error, 2649
round_style, 2651
signaling_NaN, 2649
tinyness_before, 2651
traps, 2651
std::numeric_limits< unsigned long >, 2651
denorm_min, 2653
digits, 2654
digits10, 2654
epsilon, 2653
has_denorm, 2654
has_denorm_loss, 2655
has_infinity, 2655
has_quiet_NaN, 2655
has_signaling_NaN, 2655
infinity, 2653
is_bounded, 2655
is_exact, 2655
is_iec559, 2655
is_integer, 2655
is_modulo, 2655
is_signed, 2655
is_specialized, 2656
lowest, 2654
max, 2654
max_digits10, 2656
max_exponent, 2656
max_exponent10, 2656
min, 2654
min_exponent, 2656
min_exponent10, 2656
quiet_NaN, 2654
radix, 2656
round_error, 2654
round_style, 2656
signaling_NaN, 2654
tinyness_before, 2656
traps, 2656
std::numeric_limits< unsigned long long >, 2657
denorm_min, 2659
digits, 2660
digits10, 2660
epsilon, 2659
has_denorm, 2660
has_denorm_loss, 2660
has_infinity, 2660
has_quiet_NaN, 2660
has_signaling_NaN, 2660
infinity, 2659
is_bounded, 2660
is_exact, 2660

is_iec559, 2660
is_integer, 2660
is_modulo, 2661
is_signed, 2661
is_specialized, 2661
lowest, 2659
max, 2659
max_digits10, 2661
max_exponent, 2661
max_exponent10, 2661
min, 2659
min_exponent, 2661
min_exponent10, 2661
quiet_NaN, 2659
radix, 2661
round_error, 2659
round_style, 2661
signaling_NaN, 2659
tinyness_before, 2662
traps, 2662
std::numeric_limits< unsigned short >, 2662
denorm_min, 2664
digits, 2665
digits10, 2665
epsilon, 2664
has_denorm, 2665
has_denorm_loss, 2665
has_infinity, 2665
has_quiet_NaN, 2665
has_signaling_NaN, 2665
infinity, 2664
is_bounded, 2665
is_exact, 2665
is_iec559, 2666
is_integer, 2666
is_modulo, 2666
is_signed, 2666
is_specialized, 2666
lowest, 2664
max, 2664
max_digits10, 2666
max_exponent, 2666
max_exponent10, 2666
min, 2664
min_exponent, 2666
min_exponent10, 2667
quiet_NaN, 2664
radix, 2667
round_error, 2664
round_style, 2667
signaling_NaN, 2665
tinyness_before, 2667
traps, 2667
std::numeric_limits< wchar_t >, 2667
denorm_min, 2669
digits, 2670
digits10, 2670
epsilon, 2669
has_denorm, 2670
has_denorm_loss, 2670
has_infinity, 2670
has_quiet_NaN, 2670
has_signaling_NaN, 2670
infinity, 2669
is_bounded, 2671
is_exact, 2671
is_iec559, 2671
is_integer, 2671
is_modulo, 2671
is_signed, 2671
is_specialized, 2671
lowest, 2669
max, 2669
max_digits10, 2671
max_exponent, 2671
max_exponent10, 2671
min, 2669
min_exponent, 2672
min_exponent10, 2672
quiet_NaN, 2670
radix, 2672
round_error, 2670
round_style, 2672
signaling_NaN, 2670
tinyness_before, 2672
traps, 2672
std::numpunct< _CharT >, 2672
~numpunct, 2675
char_type, 2674
decimal_point, 2675
do_decimal_point, 2675
do_falsename, 2676
do_grouping, 2676
do_thousands_sep, 2676
do_truename, 2676
falsename, 2677
grouping, 2677
id, 2678
numpunct, 2674, 2675
string_type, 2674
thousands_sep, 2677
truename, 2677
std::numpunct_byname< _CharT >, 2678
decimal_point, 2679
do_decimal_point, 2680
do_falsename, 2680
do_grouping, 2680
do_thousands_sep, 2680

do_truename, 2681
falsename, 2681
grouping, 2681
id, 2682
thousands_sep, 2681
truename, 2682
std::once_flag, 2684
call_once, 2685
once_flag, 2685
operator=, 2685
std::ostream_iterator< _Tp, _CharT, _Traits >, 2687
char_type, 2688
difference_type, 2688
iterator_category, 2688
operator=, 2689
ostream_iterator, 2689
ostream_type, 2688
pointer, 2688
reference, 2688
traits_type, 2688
value_type, 2688
std::ostreambuf_iterator< _CharT, _Traits >, 2690
char_type, 2691
difference_type, 2691
failed, 2692
iterator_category, 2691
operator++, 2693
operator=, 2693
operator*, 2693
ostream_type, 2691
ostreambuf_iterator, 2692
pointer, 2692
reference, 2692
streambuf_type, 2692
traits_type, 2692
value_type, 2692
std::out_of_range, 2693
what, 2694
std::output_iterator_tag, 2694
std::overflow_error, 2702
what, 2703
std::owner_less< _Tp >, 2703
std::owner_less< shared_ptr< _Tp > >, 2704
first_argument_type, 2705
result_type, 2705
second_argument_type, 2705
std::owner_less< void >, 2705
first_argument_type, 2706
result_type, 2706
second_argument_type, 2706
std::owner_less< weak_ptr< _Tp > >, 2707
first_argument_type, 2707
result_type, 2707
second_argument_type, 2707
std::packaged_task< _Res(_ArgTypes...)>, 2707
std::pair< _T1, _T2 >, 2711
first, 2715
first_type, 2713
pair, 2714
second, 2715
second_type, 2713
swap, 2715
std::piecewise_constant_distribution< _RealType >, 2742
densities, 2744
intervals, 2744
max, 2744
min, 2744
operator<<, 2746
operator>>, 2746
operator(), 2744
operator==, 2746
param, 2744
reset, 2746
result_type, 2744
std::piecewise_constant_distribution< _RealType >::param_type, 2727
std::piecewise_construct_t, 2747
std::piecewise_linear_distribution< _RealType >, 2747
densities, 2748
intervals, 2748
max, 2748
min, 2748
operator<<, 2749
operator>>, 2750
operator(), 2749
operator==, 2750
param, 2749
reset, 2749
result_type, 2748
std::piecewise_linear_distribution< _RealType >::param_type, 2728
std::placeholders, 708
std::plus< _Tp >, 2750
first_argument_type, 2751
operator(), 2752
result_type, 2751
second_argument_type, 2751
std::pmr::memory_resource, 2424
std::pmr::monotonic_buffer_resource, 2466
do_allocate, 2467
do_deallocate, 2467
do_is_equal, 2467
std::pmr::polymorphic_allocator< _Tp >, 2762
std::pmr::pool_options, 2763
max_blocks_per_chunk, 2764
std::pmr::unsynchronized_pool_resource, 3278
do_allocate, 3279
do_deallocate, 3279

do_is_equal, 3279
std::pointer_to_binary_function< _Arg1, _Arg2, _Result >, 2752
first_argument_type, 2753
result_type, 2753
second_argument_type, 2753
std::pointer_to_unary_function< _Arg, _Result >, 2754
argument_type, 2755
result_type, 2755
std::pointer_traits< _Ptr >, 2755
difference_type, 2755
element_type, 2755
pointer, 2755
pointer_to, 2756
rebind, 2756
std::pointer_traits< _Tp * >, 2756
difference_type, 2757
element_type, 2757
pointer, 2757
pointer_to, 2758
rebind, 2757
std::poisson_distribution< _IntType >, 2759
max, 2760
mean, 2760
min, 2760
operator<<, 2761
operator>>, 2762
operator(), 2760, 2761
operator==, 2762
param, 2761
reset, 2761
result_type, 2760
std::poisson_distribution< _IntType >::param_type, 2728
std::priority_queue< _Tp, _Sequence, _Compare >, 2765
empty, 2768
pop, 2768
priority_queue, 2767
push, 2768
size, 2769
top, 2769
std::promise< _Res >, 2771
std::promise< _Res & >, 2772
std::promise< void >, 2773
std::queue< _Tp, _Sequence >, 2775
back, 2777
c, 2778
empty, 2777
front, 2777, 2778
pop, 2778
push, 2778
queue, 2777
size, 2778
std::random_access_iterator_tag, 2780
std::random_device, 2781
result_type, 2782
std::range_error, 2782
what, 2783
std::ranges::equal_to, 2110
std::ranges::greater, 2198
std::ranges::greater_equal, 2200
std::ranges::less, 2314
std::ranges::less_equal, 2315
std::ranges::not_equal_to, 2541
std::ranges::subrange< _It, _Sent, _Kind >, 3037
std::ranges::view_interface< _Derived >, 3328
std::rank< _Tp >, 2789
std::ratio< _Num, _Den >, 2790
std::ratio_equal< _R1, _R2 >, 2790
std::ratio_greater< _R1, _R2 >, 2791
std::ratio_greater_equal< _R1, _R2 >, 2792
std::ratio_less< _R1, _R2 >, 2793
std::ratio_less_equal< _R1, _R2 >, 2794
std::ratio_not_equal< _R1, _R2 >, 2794
std::raw_storage_iterator< _OutputIterator, _Tp >, 2795
difference_type, 2796
iterator_category, 2796
pointer, 2797
reference, 2797
value_type, 2797
std::recursive_mutex, 2812
std::recursive_timed_mutex, 2812
std::reference_wrapper< _Tp >, 2814
cref, 2815
ref, 2815
std::regex_constants, 709
__multiline, 715
__polynomial, 715
awk, 715
basic, 715
collate, 715
ECMAScript, 715
egrep, 715
error_backref, 711
error_badbrace, 711
error_badrepeat, 711
error_brace, 711
error_brack, 711
error_collate, 711
error_complexity, 711
error_ctype, 711
error_escape, 712
error_paren, 712
error_range, 712
error_space, 712
error_stack, 712
error_type, 710
extended, 716
format_default, 716

format_first_only, 716
format_no_copy, 716
format_sed, 716
grep, 717
icase, 717
match_any, 717
match_continuous, 717
match_default, 717
match_flag_type, 710
match_not_bol, 717
match_not_bow, 717
match_not_eol, 717
match_not_eow, 718
match_not_null, 718
match_prev_avail, 718
multiline, 718
nosubs, 718
operator&, 712
operator&=, 712, 713
operator∼, 714
operator^, 713
operator^=, 713
operator|, 714
operator|=, 714
optimize, 718
syntax_option_type, 710
std::regex_error, 2815
code, 2816
regex_error, 2816
what, 2817
std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >, 2817
operator++, 2818, 2819
operator->, 2819
operator=, 2819
operator==, 2819
operator*, 2818
regex_iterator, 2818
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >, 2819
operator++, 2823
operator->, 2823
operator=, 2823
operator==, 2823
operator*, 2823
regex_token_iterator, 2820–2822
std::regex_traits< _Ch_type >, 2824
getloc, 2825
imbue, 2825
isctype, 2825
length, 2826
lookup_classname, 2826
lookup_collatename, 2827
regex_traits, 2825
transform, 2827
transform_primary, 2828
translate, 2828
translate_nocase, 2828
value, 2829
std::rel_ops, 718
operator!=, 719
operator<=, 719
operator>, 719
operator>=, 720
std::remove_all_extents< _Tp >, 2829
std::remove_const< _Tp >, 2830
std::remove_cv< _Tp >, 2830
std::remove_extent< _Tp >, 2830
std::remove_pointer< _Tp >, 2831
std::remove_reference< _Tp >, 2831
std::remove_volatile< _Tp >, 2831
std::result_of< _Signature >, 2833
std::reverse_iterator< _Iterator >, 2834
base, 2836
operator+, 2836
operator++, 2836
operator+=, 2837
operator-, 2837
operator->, 2838
operator--, 2837
operator-=, 2837
operator[], 2838
operator*, 2836
reverse_iterator, 2835
std::runtime_error, 2843
runtime_error, 2844
what, 2844
std::scoped_allocator_adaptor< _OuterAlloc, _InnerAllocs >, 2859
std::seed_seq, 2860
result_type, 2861
seed_seq, 2861
std::set< _Key, _Compare, _Alloc >, 2929
∼set, 2937
allocator_type, 2932
begin, 2937
cbegin, 2937
cend, 2937
clear, 2937
const_iterator, 2932
const_pointer, 2932
const_reference, 2932
const_reverse_iterator, 2932
contains, 2937, 2938
count, 2938
crbegin, 2939
crend, 2939
difference_type, 2933

emplace, 2939
emplace_hint, 2939
empty, 2940
end, 2940
equal_range, 2940, 2941
erase, 2942, 2943
find, 2943, 2944
get_allocator, 2944
insert, 2945, 2946
iterator, 2933
key_comp, 2946
key_compare, 2933
key_type, 2933
lower_bound, 2946, 2947
max_size, 2949
operator=, 2949
pointer, 2933
rbegin, 2949
reference, 2933
rend, 2950
reverse_iterator, 2933
set, 2934–2936
size, 2950
size_type, 2933
swap, 2950
upper_bound, 2950, 2951
value_comp, 2952
value_compare, 2934
value_type, 2934
std::shared_future< _Res >, 2952
_M_get_result, 2953
get, 2953
shared_future, 2953
std::shared_future< _Res & >, 2954
_M_get_result, 2955
get, 2955, 2956
shared_future, 2955
std::shared_future< void >, 2956
_M_get_result, 2958
get, 2958
shared_future, 2957
std::shared_lock< _Mutex >, 2958
std::shared_ptr< _Tp >, 2959
element_type, 2961
get, 2967
operator bool, 2967
owner_before, 2967, 2968
shared_ptr, 2961–2967
swap, 2968
unique, 2968
use_count, 2968
std::shared_timed_mutex, 2968
std::shuffle_order_engine< _RandomNumberEngine, __k >, 2969
base, 2971
discard, 2971
max, 2971
min, 2972
operator<<, 2972
operator>>, 2973
operator(), 2972
operator==, 2973
result_type, 2970
seed, 2972
shuffle_order_engine, 2970, 2971
std::slice, 2974
operator==, 2974
std::slice_array< _Tp >, 2974
operator<<=, 2976
operator>>=, 2976
operator+=, 2976
operator-=, 2976
operator/=, 2976
operator%=, 2976
operator&=, 2976
operator*=, 2976
operator^=, 2977
operator|=, 2977
std::stack< _Tp, _Sequence >, 2985
empty, 2987
pop, 2987
push, 2987
size, 2987
stack, 2987
top, 2987, 2988
std::student_t_distribution< _RealType >, 3031
max, 3032
min, 3032
operator<<, 3033
operator>>, 3034
operator(), 3032
operator==, 3033
param, 3033
reset, 3033
result_type, 3032
std::student_t_distribution< _RealType >::param_type, 2729
std::sub_match< _BiIter >, 3034
compare, 3036
length, 3037
operator string_type, 3037
str, 3037
swap, 3037
std::subtract_with_carry_engine< _UIntType, __w, __s, __r >, 3039
discard, 3041
max, 3041
min, 3041

operator<<, 3042
operator>>, 3042
operator(), 3041
operator==, 3042
result_type, 3040
seed, 3041
subtract_with_carry_engine, 3040
std::system_error, 3046
what, 3046
std::this_thread, 720
get_id, 721
sleep_for, 721
sleep_until, 721
yield, 721
std::thread, 3052
native_handle, 3053
std::thread::id, 2240
std::time_base, 3061
std::time_get< _CharT, _InIter >, 3062
~time_get, 3064
char_type, 3064
date_order, 3064
do_date_order, 3064
do_get, 3065
do_get_date, 3065
do_get_monthname, 3066
do_get_time, 3067
do_get_weekday, 3067
do_get_year, 3068
get, 3068, 3069
get_date, 3070
get_monthname, 3070
get_time, 3071
get_weekday, 3071
get_year, 3072
id, 3073
iter_type, 3064
time_get, 3064
std::time_get_byname< _CharT, _InIter >, 3073
date_order, 3075
do_date_order, 3075
do_get, 3075
do_get_date, 3076
do_get_monthname, 3076
do_get_time, 3077
do_get_weekday, 3078
do_get_year, 3078
get, 3079
get_date, 3080
get_monthname, 3081
get_time, 3081
get_weekday, 3082
get_year, 3082
id, 3083
std::time_put< _CharT, _OutIter >, 3084
~time_put, 3086
char_type, 3086
do_put, 3086
id, 3088
iter_type, 3086
put, 3087, 3088
time_put, 3086
std::time_put_byname< _CharT, _OutIter >, 3088
do_put, 3090
id, 3091
put, 3090, 3091
std::timed_mutex, 3092
std::to_chars_result, 3092
std::tr1, 721
arg, 724
conf_hyperg, 724
conf_hypergf, 725
conf_hypergl, 725
hyperg, 725
hypergf, 726
hypergl, 726
norm, 726
polar, 726
std::tr1::__detail, 727
std::tr2, 727
std::tr2::__detail, 728
std::tr2::__dynamic_bitset_base< _WordT, _Alloc >, 762
_M_w, 764
std::tr2::__reflection_typelist< _Elements >, 799
std::tr2::__reflection_typelist< _First, _Rest... >, 799
std::tr2::__reflection_typelist<>, 800
std::tr2::bases< _Tp >, 1088
std::tr2::bool_set, 1770
bool_set, 1771
equals, 1771
is_emptyset, 1771
is_indeterminate, 1771
is_singleton, 1771
operator bool, 1771
std::tr2::direct_bases< _Tp >, 2051
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2069
all, 2074
any, 2074
append, 2074, 2075
clear, 2075
count, 2075
dynamic_bitset, 2073, 2074
empty, 2075
find_first, 2075
find_next, 2075
flip, 2076
get_allocator, 2076
max_size, 2076

none, 2076
num_blocks, 2076
operator<<, 2077
operator<<=, 2077
operator>>, 2078
operator>>=, 2078
operator-=, 2077
operator=, 2078
operator&=, 2077
operator[], 2078, 2079
operator~, 2080
operator^=, 2079
operator|=, 2079
push_back, 2080
reset, 2080
resize, 2080
set, 2080, 2081
size, 2081
swap, 2081
test, 2081
to_string, 2082
to_ullong, 2082
to_ulong, 2082
std::tr2::dynamic_bitset< _WordT, _Alloc >::reference, 2813
std::try_to_lock_t, 3123
std::tuple< _Elements >, 3123
std::tuple< _T1, _T2 >, 3125
std::tuple_element< 0, pair< _Tp1, _Tp2 > >, 3128
std::tuple_element< 1, pair< _Tp1, _Tp2 > >, 3128
std::tuple_element< __i, _Tp >, 3127
std::tuple_element< __i, tuple< _Types... > >, 3128
std::tuple_element< _Ind, array< _Tp, _Nm > >, 3128
std::tuple_size< _Tp >, 3129
std::tuple_size< array< _Tp, _Nm > >, 3129
std::tuple_size< pair< _Tp1, _Tp2 > >, 3130
std::tuple_size< tuple< _Elements... > >, 3131
std::type_index, 3132
std::type_info, 3133
~type_info, 3134
before, 3134
name, 3134
std::unary_function< _Arg, _Result >, 3139
argument_type, 3139
result_type, 3139
std::unary_negate< _Predicate >, 3140
argument_type, 3140
result_type, 3140
std::underflow_error, 3142
what, 3142
std::underlying_type< _Tp >, 3143
std::uniform_int_distribution< _IntType >, 3143
max, 3144
min, 3144
operator(), 3144
operator==, 3145
param, 3145
reset, 3145
result_type, 3144
uniform_int_distribution, 3144
std::uniform_int_distribution< _IntType >::param_type, 2729
std::uniform_real_distribution< _RealType >, 3145
max, 3147
min, 3147
operator(), 3147
operator==, 3148
param, 3147
reset, 3148
result_type, 3146
uniform_real_distribution, 3146
std::uniform_real_distribution< _RealType >::param_type, 2730
std::unique_lock< _Mutex >, 3148
swap, 3149
std::unique_ptr< _Tp, _Dp >, 3149
~unique_ptr, 3152
get, 3152
get_deleter, 3153
operator bool, 3153
operator->, 3153
operator=, 3153
operator*, 3153
release, 3154
reset, 3154
swap, 3154
unique_ptr, 3151, 3152
std::unique_ptr< _Tp[], _Dp >, 3154
~unique_ptr, 3158, 3160
get, 3160
get_deleter, 3160
operator bool, 3160
operator<<, 3163
operator->, 3161
operator=, 3161, 3162
operator[], 3162
operator*, 3161
release, 3162
reset, 3162, 3163
swap, 3163, 3164
unique_ptr, 3157–3159
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3167
allocator_type, 3171
at, 3175
begin, 3176
bucket_count, 3177
cbegin, 3177

cend, 3177
clear, 3178
const_iterator, 3171
const_local_iterator, 3171
const_pointer, 3171
const_reference, 3171
contains, 3178
count, 3179
difference_type, 3171
emplace, 3179
emplace_hint, 3181
empty, 3181
end, 3181, 3182
equal_range, 3182, 3183
erase, 3184, 3185
find, 3185, 3186
get_allocator, 3187
hash_function, 3187
hasher, 3172
insert, 3187–3190
iterator, 3172
key_eq, 3190
key_equal, 3172
key_type, 3172
load_factor, 3190
local_iterator, 3172
mapped_type, 3172
max_bucket_count, 3191
max_load_factor, 3191
max_size, 3191
operator=, 3191, 3192
operator[], 3192
pointer, 3172
reference, 3172
rehash, 3193
reserve, 3193
size, 3193
size_type, 3173
swap, 3193
unordered_map, 3173, 3174
value_type, 3173
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3198
allocator_type, 3201
begin, 3206
bucket_count, 3207
cbegin, 3207
cend, 3207
clear, 3208
const_iterator, 3201
const_local_iterator, 3202
const_pointer, 3202
const_reference, 3202
contains, 3208
count, 3209
difference_type, 3202
emplace, 3209
emplace_hint, 3210
empty, 3210
end, 3210, 3211
equal_range, 3211, 3212
erase, 3213, 3214
find, 3214, 3215
get_allocator, 3216
hash_function, 3216
hasher, 3202
insert, 3216, 3217, 3219
iterator, 3202
key_eq, 3220
key_equal, 3202
key_type, 3203
load_factor, 3220
local_iterator, 3203
mapped_type, 3203
max_bucket_count, 3220
max_load_factor, 3220
max_size, 3222
operator=, 3222
pointer, 3203
reference, 3203
rehash, 3222
reserve, 3223
size, 3223
size_type, 3203
swap, 3223
unordered_multimap, 3204, 3205
value_type, 3203
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3226
allocator_type, 3230
begin, 3234
bucket_count, 3235
cbegin, 3235
cend, 3235
clear, 3236
const_iterator, 3230
const_local_iterator, 3230
const_pointer, 3230
const_reference, 3230
contains, 3236
count, 3237
difference_type, 3230
emplace, 3237
emplace_hint, 3238
empty, 3238
end, 3238, 3239
equal_range, 3239, 3240
erase, 3240–3242

find, 3242, 3244
get_allocator, 3244
hash_function, 3245
hasher, 3231
insert, 3245–3247
iterator, 3231
key_eq, 3247
key_equal, 3231
key_type, 3231
load_factor, 3247
local_iterator, 3231
max_bucket_count, 3247
max_load_factor, 3247, 3248
max_size, 3248
operator=, 3248, 3249
pointer, 3231
reference, 3231
rehash, 3249
reserve, 3249
size, 3249
size_type, 3232
swap, 3249
unordered_multiset, 3232, 3233
value_type, 3232
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3253
allocator_type, 3256
begin, 3260, 3262
bucket_count, 3262
cbegin, 3262
cend, 3263
clear, 3263
const_iterator, 3256
const_local_iterator, 3256
const_pointer, 3256
const_reference, 3257
contains, 3263
count, 3264
difference_type, 3257
emplace, 3264
emplace_hint, 3265
empty, 3265
end, 3265, 3266
equal_range, 3266, 3268
erase, 3270, 3271
find, 3271, 3272
get_allocator, 3273
hash_function, 3273
hasher, 3257
insert, 3273–3275
iterator, 3257
key_eq, 3275
key_equal, 3257
key_type, 3257
load_factor, 3275
local_iterator, 3257
max_bucket_count, 3276
max_load_factor, 3276
max_size, 3276
operator=, 3276, 3277
pointer, 3258
reference, 3258
rehash, 3277
reserve, 3277
size, 3278
size_type, 3258
swap, 3278
unordered_set, 3258–3260
value_type, 3258
std::uses_allocator< tuple< _Types... >, _Alloc >, 3280
std::uses_allocator< typename, typename >, 3280
std::valarray< _Tp >, 3281
operator!, 3283
operator<<=, 3285
operator>>=, 3285
operator+, 3284
operator+=, 3284
operator-, 3284
operator-=, 3284
operator/=, 3285
operator%=, 3283
operator&=, 3283
operator*=, 3284
operator~, 3286
operator^=, 3285
operator|=, 3286
valarray, 3283
std::vector< _Tp, _Alloc >, 3289
_M_allocate_and_copy, 3295
_M_range_check, 3295
~vector, 3295
assign, 3295, 3296
at, 3296, 3297
back, 3297
begin, 3298
capacity, 3298
cbegin, 3298
cend, 3298
clear, 3298
crbegin, 3299
crend, 3299
data, 3299
emplace, 3299
empty, 3299
end, 3299, 3300
erase, 3300
front, 3301
get_allocator, 3301

insert, 3301–3303
max_size, 3303
operator=, 3303, 3304
operator[], 3304
pop_back, 3305
push_back, 3305
rbegin, 3305
rend, 3306
reserve, 3306
resize, 3306, 3307
shrink_to_fit, 3307
size, 3307
swap, 3307
vector, 3293–3295
std::vector< bool, _Alloc >, 3308
_M_allocate_and_copy, 3316
_M_range_check, 3316
~vector, 3316
assign, 3316, 3318
at, 3318
back, 3319
begin, 3319
capacity, 3319
cbegin, 3319
cend, 3319
clear, 3319
crbegin, 3320
crend, 3320
data, 3320
emplace, 3320
empty, 3320
end, 3320
erase, 3321
front, 3321
get_allocator, 3321, 3322
insert, 3322, 3323
max_size, 3323
operator=, 3324
operator[], 3324, 3325
pop_back, 3325
push_back, 3325
rbegin, 3325, 3326
rend, 3326
reserve, 3326
resize, 3326, 3327
shrink_to_fit, 3327
size, 3327
swap, 3327
vector, 3313–3315
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3329
_M_buf_locale, 3344
_M_in_beg, 3344
_M_in_cur, 3344
_M_in_end, 3344
_M_out_beg, 3344
_M_out_cur, 3344
_M_out_end, 3344
__streambuf_type, 3331
char_type, 3331
eback, 3332
egptr, 3332
epptr, 3332
gbump, 3333
getloc, 3333
gptr, 3333
imbue, 3334
in_avail, 3334
int_type, 3331
off_type, 3331
overflow, 3334
pbackfail, 3335
pbase, 3335
pbump, 3335
pos_type, 3331
pptr, 3336
pubimbue, 3336
pubseekoff, 3336
pubseekpos, 3337
pubsetbuf, 3337
pubsync, 3337
sbumpc, 3337
seekoff, 3337
seekpos, 3338
setbuf, 3338
setg, 3338
setp, 3339
sgetc, 3339
sgetn, 3339
showmanyc, 3340
snextc, 3340
sputbackc, 3340
sputc, 3341
sputn, 3341
state, 3342
sungetc, 3342
sync, 3342
traits_type, 3331
uflow, 3342
underflow, 3342
wbuffer_convert, 3331, 3332
xsgetn, 3343
xsputn, 3343
std::weak_ptr< _Tp >, 3345
std::weibull_distribution< _RealType >, 3346
a, 3347
b, 3347
max, 3347
min, 3347

operator(), 3347
operator==, 3348
param, 3348
reset, 3348
result_type, 3347
std::weibull_distribution< _RealType >::param_type, 2730
std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >, 3348
converted, 3350
from_bytes, 3350, 3351
state, 3351
to_bytes, 3351, 3352
wstring_convert, 3349, 3350
std_abs.h, 3466
std_function.h, 3466
std_mutex.h, 3467
std_thread.h, 3467
stdatomic.h, 3756
stdbit.h, 3756
stdc++.h, 3800
stdckdint.h, 3756
stddev
std::normal_distribution< _RealType >, 2538
stdexcept, 3756
stdio_filebuf
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2991
stdio_filebuf.h, 3672
stdio_sync_filebuf.h, 3672
stdlib.h, 3757
abs, 3757
stdtr1c++.h, 3800
stl_algo.h, 3468
stl_algobase.h, 3477
stl_bvector.h, 3482
stl_construct.h, 3483
stl_deque.h, 3484
_GLIBCXX_DEQUE_BUF_SIZE, 3484
stl_function.h, 3485
stl_heap.h, 3486
stl_iterator.h, 3488, 3490
stl_iterator_base_funcs.h, 3491
stl_iterator_base_types.h, 3492
stl_list.h, 3493
stl_map.h, 3494
stl_multimap.h, 3495
stl_multiset.h, 3496
stl_numeric.h, 3497
stl_pair.h, 3497
stl_queue.h, 3498
stl_raw_storage_iter.h, 3500
stl_relops.h, 3500
stl_set.h, 3500
stl_stack.h, 3501
stl_tempbuf.h, 3502
stl_tree.h, 3503
stl_uninitialized.h, 3503
stl_vector.h, 3504
stop_token, 3757
str
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1340
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1437
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1664, 1665
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1707, 1708
std::match_results< _Bi_iter, _Alloc >, 2416
std::sub_match< _BiIter >, 3037
stream_iterator.h, 3505
streambuf, 3757
I/O, 165
streambuf.tcc, 3505
streambuf_iterator.h, 3505
streambuf_type
std::istreambuf_iterator< _CharT, _Traits >, 2295
std::ostreambuf_iterator< _CharT, _Traits >, 2692
streamoff
std, 580
streampos
std, 580
streamsize
std, 580
stride
Numeric Arrays, 232
string, 3758, 3760, 3761
Strings, 273
string_conversions.h, 3673
string_type
std::__cxx11::collate< _CharT >, 1857
std::__cxx11::collate_byname< _CharT >, 1862
std::messages< _CharT >, 2431
std::money_get< _CharT, _InIter >, 2442
std::money_put< _CharT, _OutIter >, 2446
std::moneypunct< _CharT, _Intl >, 2451
std::numpunct< _CharT >, 2674
string_view, 3761, 3763
string_view.tcc, 3506
stringbuf
I/O, 165
stringfwd.h, 3507
Strings, 273
string, 273
u16string, 273
u32string, 273
wstring, 273
stringstream

I/O, 166
strstream, 3363
substr
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 857
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1544
std::basic_string< _CharT, _Traits, _Alloc >, 1642, 1643
subtract_with_carry_engine
std::subtract_with_carry_engine< _UIntType, __w, __s, __r >, 3040
subtractive_rng
__gnu_cxx::subtractive_rng, 3044
suffix
std::match_results< _Bi_iter, _Alloc >, 2417
sum
Numeric Arrays, 232
sungetc
__gnu_cxx::enc_filebuf< _CharT >, 2097
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 3005
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3023
std::basic_filebuf< _CharT, _Traits >, 1105
std::basic_streambuf< _CharT, _Traits >, 1470
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1665
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3342
swap
__gnu_cxx, 403
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 857
__gnu_cxx::pair< _T1, _T2 >, 2711
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1544
__gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >, 896, 898
__gnu_pbds::sample_probe_fn, 2845
__gnu_pbds::sample_range_hashing, 2846
__gnu_pbds::sample_ranged_hash_fn, 2847
__gnu_pbds::sample_resize_policy, 2850
__gnu_pbds::sample_resize_trigger, 2853
__gnu_pbds::sample_size_policy, 2854
__gnu_pbds::sample_update_policy, 2858
Futures, 115
Metaprogramming, 353
Numeric Arrays, 232
Pointer Abstractions, 342
Regular Expressions, 271
std, 642–644
std::__debug, 654
std::basic_regex< _Ch_type, _Rx_traits >, 1455
std::basic_string< _CharT, _Traits, _Alloc >, 1643
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>, 1908
std::deque< _Tp, _Alloc >, 2050
std::experimental::fundamentals_v1::any, 1066
std::forward_list< _Tp, _Alloc >, 2156
std::function< _Res(_ArgTypes...)>, 2168
std::list< _Tp, _Alloc >, 2346
std::map< _Key, _Tp, _Compare, _Alloc >, 2403
std::match_results< _Bi_iter, _Alloc >, 2417
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>, 2472
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2497
std::multiset< _Key, _Compare, _Alloc >, 2522
std::pair< _T1, _T2 >, 2715
std::set< _Key, _Compare, _Alloc >, 2950
std::shared_ptr< _Tp >, 2968
std::sub_match< _BiIter >, 3037
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2081
std::unique_lock< _Mutex >, 3149
std::unique_ptr< _Tp, _Dp >, 3154
std::unique_ptr< _Tp[], _Dp >, 3163, 3164
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3193
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3223
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3249
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3278
std::vector< _Tp, _Alloc >, 3307
std::vector< bool, _Alloc >, 3327
Type-safe container of any type, 290
Utilities, 304, 305
swap_ranges
Mutating, 37
sync
__gnu_cxx::enc_filebuf< _CharT >, 2097
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 3005
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3023
std::basic_filebuf< _CharT, _Traits >, 1105
std::basic_fstream< _CharT, _Traits >, 1151
std::basic_ifstream< _CharT, _Traits >, 1194
std::basic_iostream< _CharT, _Traits >, 1262
std::basic_istream< _CharT, _Traits >, 1301
std::basic_istream< _CharT, _Traits >::sentry, 2891
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1340
std::basic_streambuf< _CharT, _Traits >, 1470
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1665
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1708
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3342
sync_with_stdio
std::basic_fstream< _CharT, _Traits >, 1151

std::basic_ifstream< _CharT, _Traits >, 1194
std::basic_ios< _CharT, _Traits >, 1218
std::basic_iostream< _CharT, _Traits >, 1263
std::basic_istream< _CharT, _Traits >, 1301
std::basic_istream< _CharT, _Traits >::sentry, 2891
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1340
std::basic_ofstream< _CharT, _Traits >, 1375
std::basic_ostream< _CharT, _Traits >, 1405
std::basic_ostream< _CharT, _Traits >::sentry, 2919
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1437
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1708
std::ios_base, 2264
syncstream, 3764
syntax_option_type
std::regex_constants, 710
synth_access_traits
__gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >, 3121
__gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >, 3123
synth_access_traits.hpp, 3645
system_category
Diagnostics, 126
system_error, 3764, 3765

t
std::binomial_distribution< _IntType >, 1750
table
std::ctype< char >, 1951
std::ctype_byname< char >, 2008
table_size
std::ctype< char >, 1958
std::ctype_byname< char >, 2013
tag_and_trait.hpp, 3656
Tags, 144
trivial_iterator_difference_type, 144
tags.h, 3746
tan
Complex Numbers, 187
math.h, 3703
tanh
Complex Numbers, 187
math.h, 3703
target
std::function< _Res(_ArgTypes...)>, 2169
target_type
std::function< _Res(_ArgTypes...)>, 2169
Technical Specifications, 274
tellg
std::basic_fstream< _CharT, _Traits >, 1151
std::basic_ifstream< _CharT, _Traits >, 1195
std::basic_iostream< _CharT, _Traits >, 1263
std::basic_istream< _CharT, _Traits >, 1301
std::basic_istream< _CharT, _Traits >::sentry, 2891
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1341
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1709
tellp
std::basic_fstream< _CharT, _Traits >, 1152
std::basic_iostream< _CharT, _Traits >, 1263
std::basic_ofstream< _CharT, _Traits >, 1375
std::basic_ostream< _CharT, _Traits >, 1405
std::basic_ostream< _CharT, _Traits >::sentry, 2919
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1439
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1709
temporary_buffer
__gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >, 3048
terminate
Exceptions, 131
terminate_handler
Exceptions, 128
test
std::bitset< _Nb >, 1769
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2081
text_encoding, 3765
text_encoding-data.h, 3507
tgmath.h, 3765
thin_heap_.hpp, 3650
this_thread_sleep.h, 3507
thousands_sep
std::moneypunct< _CharT, _Intl >, 2458
std::moneypunct_byname< _CharT, _Intl >, 2465
std::numpunct< _CharT >, 2677
std::numpunct_byname< _CharT >, 2681
thread, 3766
Threads, 119
throw_allocator.h, 3673
throw_with_nested
Exceptions, 131
tie
std::basic_fstream< _CharT, _Traits >, 1152
std::basic_ifstream< _CharT, _Traits >, 1195
std::basic_ios< _CharT, _Traits >, 1218
std::basic_iostream< _CharT, _Traits >, 1264
std::basic_istream< _CharT, _Traits >, 1302
std::basic_istream< _CharT, _Traits >::sentry, 2892
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1341
std::basic_ofstream< _CharT, _Traits >, 1375

std::basic_ostream< _CharT, _Traits >, 1405
std::basic_ostream< _CharT, _Traits >::sentry, 2919
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1439
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1709
Utilities, 305
Time, 356
abs, 366
ceil, 366, 367
clock_cast, 367
days, 365
duration_cast, 367
floor, 368
high_resolution_clock, 365
hours, 365
local_time_format, 369
microseconds, 365
milliseconds, 365
minutes, 365
months, 365
nanoseconds, 365
operator<, 373
operator<<, 373
operator<=, 374
operator<=>, 374
operator>, 374
operator>=, 374
operator+, 372
operator-, 372
operator/, 373
operator==, 374
operator%, 371
operator*, 371
round, 374, 376
seconds, 365
time_point_cast, 376
weeks, 366
years, 366
time
std::locale, 2358
time_get
std::time_get< _CharT, _InIter >, 3064
time_members.h, 3801
time_point_cast
Time, 376
time_put
std::time_put< _CharT, _OutIter >, 3086
tinyness_before
std::__numeric_limits_base, 790
std::numeric_limits< _Tp >, 2578
std::numeric_limits< bool >, 2583
std::numeric_limits< char >, 2588
std::numeric_limits< char16_t >, 2593
std::numeric_limits< char32_t >, 2599
std::numeric_limits< double >, 2604
std::numeric_limits< float >, 2609
std::numeric_limits< int >, 2614
std::numeric_limits< long >, 2620
std::numeric_limits< long double >, 2625
std::numeric_limits< long long >, 2630
std::numeric_limits< short >, 2635
std::numeric_limits< signed char >, 2641
std::numeric_limits< unsigned char >, 2646
std::numeric_limits< unsigned int >, 2651
std::numeric_limits< unsigned long >, 2656
std::numeric_limits< unsigned long long >, 2662
std::numeric_limits< unsigned short >, 2667
std::numeric_limits< wchar_t >, 2672
TLB_size
__gnu_parallel::_Settings, 1015
to_address
Pointer Abstractions, 342, 343
to_array
Array creation functions, 282
to_bytes
std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >, 3351, 3352
to_string
std::bitset< _Nb >, 1769
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2082
to_ullong
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2082
to_ulong
std::bitset< _Nb >, 1770
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2082
Todo List, 2
tolower
std, 645
std::__ctype_abstract_base< _CharT >, 759, 760
std::ctype< _CharT >, 1920, 1921
std::ctype< char >, 1951–1953
std::ctype< wchar_t >, 1974
std::ctype_byname< _CharT >, 1988
std::ctype_byname< char >, 2008, 2009
top
std::priority_queue< _Tp, _Sequence, _Compare >, 2769
std::stack< _Tp, _Sequence >, 2987, 2988
toupper
std, 645
std::__ctype_abstract_base< _CharT >, 760
std::ctype< _CharT >, 1921
std::ctype< char >, 1953–1955
std::ctype< wchar_t >, 1975
std::ctype_byname< _CharT >, 1988, 1989
std::ctype_byname< char >, 2010, 2011
TR1 Mathematical Special Functions, 246

assoc_laguerre, 248
assoc_legendre, 248
beta, 249
comp_ellint_1, 249
comp_ellint_2, 249
comp_ellint_3, 249
cyl_bessel_i, 249
cyl_bessel_j, 249
cyl_bessel_k, 249
cyl_neumann, 250
ellint_1, 250
ellint_2, 250
ellint_3, 250
expint, 250
hermite, 250
laguerre, 250
legendre, 251
riemann_zeta, 251
sph_bessel, 251
sph_legendre, 251
sph_neumann, 251
trace_fn_imps.hpp, 3628
Traits, 146
traits.hpp, 3613, 3614
traits_type
std::basic_ios< _CharT, _Traits >, 1208
std::basic_istream< _CharT, _Traits >::sentry, 2870
std::basic_streambuf< _CharT, _Traits >, 1460
std::istreambuf_iterator< _CharT, _Traits >, 2295
std::ostream_iterator< _Tp, _CharT, _Traits >, 2688
std::ostreambuf_iterator< _CharT, _Traits >, 2692
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3331
transform
Mutating, 37, 38
std::__cxx11::collate< _CharT >, 1860
std::regex_traits< _Ch_type >, 2827
transform_exclusive_scan
Generalized Numeric operations, 15
transform_inclusive_scan
Generalized Numeric operations, 15, 16
transform_minimal_n
__gnu_parallel::_Settings, 1015
transform_primary
std::regex_traits< _Ch_type >, 2828
transform_reduce
Generalized Numeric operations, 17, 18
translate
std::regex_traits< _Ch_type >, 2828
translate_nocase
std::regex_traits< _Ch_type >, 2828
traps
std::__numeric_limits_base, 790
std::numeric_limits< _Tp >, 2578
std::numeric_limits< bool >, 2583
std::numeric_limits< char >, 2588
std::numeric_limits< char16_t >, 2593
std::numeric_limits< char32_t >, 2599
std::numeric_limits< double >, 2604
std::numeric_limits< float >, 2609
std::numeric_limits< int >, 2614
std::numeric_limits< long >, 2620
std::numeric_limits< long double >, 2625
std::numeric_limits< long long >, 2630
std::numeric_limits< short >, 2635
std::numeric_limits< signed char >, 2641
std::numeric_limits< unsigned char >, 2646
std::numeric_limits< unsigned int >, 2651
std::numeric_limits< unsigned long >, 2656
std::numeric_limits< unsigned long long >, 2662
std::numeric_limits< unsigned short >, 2667
std::numeric_limits< wchar_t >, 2672
tree
__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >, 3095
tree_policy.hpp, 3657
tree_trace_base.hpp, 3651
trie
__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >, 3108
trie_policy.hpp, 3657
trie_policy_base.hpp, 3652
trie_string_access_traits_imp.hpp, 3652
trivial_iterator_difference_type
Tags, 144
true_type
Metaprogramming, 352
truename
std::numpunct< _CharT >, 2677
std::numpunct_byname< _CharT >, 2682
trunc
std::basic_fstream< _CharT, _Traits >, 1159
std::basic_ifstream< _CharT, _Traits >, 1201
std::basic_ios< _CharT, _Traits >, 1224
std::basic_iostream< _CharT, _Traits >, 1271
std::basic_istream< _CharT, _Traits >, 1308
std::basic_istream< _CharT, _Traits >::sentry, 2897
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1347
std::basic_ofstream< _CharT, _Traits >, 1382
std::basic_ostream< _CharT, _Traits >, 1412
std::basic_ostream< _CharT, _Traits >::sentry, 2925
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1445
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1716
std::ios_base, 2270
try_lock
Mutexes, 117

try_to_lock
Mutexes, 118
tuple, 3768, 3769
tuple_cat
Utilities, 306
type
__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, cc_hash_tag, Policy_Tl >, 1890
__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, gp_hash_tag, Policy_Tl >, 1891
__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, list_update_tag, Policy_Tl >, 1891
__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, ov_tree_tag, Policy_Tl >, 1892
__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, rb_tree_tag, Policy_Tl >, 1892
__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, splay_tree_tag, Policy_Tl >, 1893
__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, cc_hash_tag, Policy_Tl >, 1893
__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, gp_hash_tag, Policy_Tl >, 1894
__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, list_update_tag, Policy_Tl >, 1894
__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, ov_tree_tag, Policy_Tl >, 1895
__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, pat_trie_tag, Policy_Tl >, 1895
__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, splay_tree_tag, Policy_Tl >, 1896
__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binary_heap_tag, null_type >, 1888
__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binomial_heap_tag, null_type >, 1888
__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, pairing_heap_tag, null_type >, 1889
__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type >, 1889
__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, thin_heap_tag, null_type >, 1890
__gnu_pbds::detail::default_comb_hash_fn, 2019
__gnu_pbds::detail::default_eq_fn< Key >, 2022
__gnu_pbds::detail::default_hash_fn< Key >, 2022
__gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn >, 2024
__gnu_pbds::detail::default_resize_policy< Comb_Hash_Fn >, 2024
__gnu_pbds::detail::default_trie_access_traits< std::basic_string< Char, Char_Traits, std::allocator< char > > >, 2025
__gnu_pbds::detail::default_update_policy, 2026
__gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, true >, 2106
Metaprogramming, 352
std::experimental::fundamentals_v1::any, 1066
Type-safe container of any type, 287
any_cast, 288–290
swap, 290
type_traits, 3771, 3774, 3775
type_traits.h, 3674
type_utils.hpp, 3652
typeindex, 3779
typeinfo, 3806
typelist.h, 3674
types.h, 3747
types_traits.hpp, 3653

u16streampos
std, 580
u16string
__gnu_debug, 411
Strings, 273
u32streampos
std, 580
u32string
__gnu_debug, 411
Strings, 273
u8path
Filesystem, 162, 163
uflow
__gnu_cxx::enc_filebuf< _CharT >, 2098
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 3005
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3024
std::basic_filebuf< _CharT, _Traits >, 1106
std::basic_streambuf< _CharT, _Traits >, 1471
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1665
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3342
uncaught_exception
Exceptions, 131
undeclare_no_pointers
Pointer Safety and Garbage Collection, 344
undeclare_reachable

Pointer Safety and Garbage Collection, 344
underflow
__gnu_cxx::enc_filebuf< _CharT >, 2098
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 3005
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3024
std::basic_filebuf< _CharT, _Traits >, 1106
std::basic_streambuf< _CharT, _Traits >, 1471
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1666
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3342
underlying_type_t
Metaprogramming, 352
unexpected
Exceptions, 132
unexpected_handler
Exceptions, 128
unget
std::basic_fstream< _CharT, _Traits >, 1153
std::basic_ifstream< _CharT, _Traits >, 1196
std::basic_iostream< _CharT, _Traits >, 1264
std::basic_istream< _CharT, _Traits >, 1302
std::basic_istream< _CharT, _Traits >::sentry, 2892
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1342
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1710
unicode-data.h, 3508
unicode.h, 3508
Uniform Distributions, 242
operator<<, 243
operator>>, 243, 244
uniform_int_dist.h, 3509
uniform_int_distribution
std::uniform_int_distribution< _IntType >, 3144
uniform_real_distribution
std::uniform_real_distribution< _RealType >, 3146
uninitialized_copy
Memory, 319
uninitialized_copy_n
Memory, 319
SGI, 156
uninitialized_default_construct
Memory, 320
uninitialized_default_construct_n
Memory, 320
uninitialized_fill
Memory, 320
uninitialized_fill_n
Memory, 321
uninitialized_move
Memory, 321
uninitialized_move_n
Memory, 322
uninitialized_value_construct
Memory, 322
uninitialized_value_construct_n
Memory, 322
unique
Mutating, 38, 40
std::forward_list< _Tp, _Alloc >, 2157
std::list< _Tp, _Alloc >, 2347
std::shared_ptr< _Tp >, 2968
unique_copy
Mutating, 40, 41
unique_copy.h, 3747
unique_copy_minimal_n
__gnu_parallel::_Settings, 1015
unique_lock.h, 3510
unique_ptr
std::unique_ptr< _Tp, _Dp >, 3151, 3152
std::unique_ptr< _Tp[], _Dp >, 3157–3159
unique_ptr.h, 3510
unitbuf
std, 645
std::basic_fstream< _CharT, _Traits >, 1159
std::basic_ifstream< _CharT, _Traits >, 1202
std::basic_ios< _CharT, _Traits >, 1224
std::basic_iostream< _CharT, _Traits >, 1271
std::basic_istream< _CharT, _Traits >, 1308
std::basic_istream< _CharT, _Traits >::sentry, 2897
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1348
std::basic_ofstream< _CharT, _Traits >, 1382
std::basic_ostream< _CharT, _Traits >, 1412
std::basic_ostream< _CharT, _Traits >::sentry, 2925
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1446
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1716
std::ios_base, 2270
Unordered Associative, 122
unordered_map, 3779, 3781, 3782
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3173, 3174
unordered_map.h, 3511
unordered_multimap
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3204, 3205
unordered_multiset
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3232, 3233
unordered_set, 3782, 3784, 3785
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3258–3260
unordered_set.h, 3513
unsetf
std::basic_fstream< _CharT, _Traits >, 1153
std::basic_ifstream< _CharT, _Traits >, 1196

std::basic_ios< _CharT, _Traits >, 1219
std::basic_iostream< _CharT, _Traits >, 1265
std::basic_istream< _CharT, _Traits >, 1303
std::basic_istream< _CharT, _Traits >::sentry, 2892
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1342
std::basic_ofstream< _CharT, _Traits >, 1376
std::basic_ostream< _CharT, _Traits >, 1406
std::basic_ostream< _CharT, _Traits >::sentry, 2920
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1439
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1710
std::ios_base, 2265
unshift
std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >, 745
std::codecvt< _InternT, _ExternT, _StateT >, 1804
std::codecvt< _InternT, _ExternT, encoding_state >, 1812, 1813
std::codecvt< char, char, mbstate_t >, 1821, 1822
std::codecvt< char16_t, char, mbstate_t >, 1830
std::codecvt< char32_t, char, mbstate_t >, 1839
std::codecvt< wchar_t, char, mbstate_t >, 1848, 1849
std::codecvt_byname< _InternT, _ExternT, _StateT >, 1855
update_fn_imps.hpp, 3645
upper_bound
Binary Search, 85
std::map< _Key, _Tp, _Compare, _Alloc >, 2403, 2404
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2497, 2498
std::multiset< _Key, _Compare, _Alloc >, 2522, 2523
std::set< _Key, _Compare, _Alloc >, 2950, 2951
uppercase
std, 645
std::basic_fstream< _CharT, _Traits >, 1159
std::basic_ifstream< _CharT, _Traits >, 1202
std::basic_ios< _CharT, _Traits >, 1224
std::basic_iostream< _CharT, _Traits >, 1271
std::basic_istream< _CharT, _Traits >, 1308
std::basic_istream< _CharT, _Traits >::sentry, 2897
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1348
std::basic_ofstream< _CharT, _Traits >, 1382
std::basic_ostream< _CharT, _Traits >, 1412
std::basic_ostream< _CharT, _Traits >::sentry, 2925
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1446
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1716
std::ios_base, 2270
use_count
std::shared_ptr< _Tp >, 2968
use_facet
Locales, 175
std::locale, 2356
std::locale::id, 2239
uses_allocator_args.h, 3516
Utilities, 295
__addressof, 298
__invoke, 299
addressof, 299
declval, 299
forward, 300
forward_as_tuple, 301
get, 301
make_pair, 301
make_tuple, 302
move, 302
move_if_noexcept, 303
operator<=>, 304
operator==, 304
pair, 298
piecewise_construct, 306
swap, 304, 305
tie, 305
tuple_cat, 306
utility, 3786
utility.h, 3516

valarray, 3786
Numeric Arrays, 220, 221
std::valarray< _Tp >, 3283
valarray_after.h, 3517
valarray_array.h, 3527
valarray_array.tcc, 3535
valarray_before.h, 3535
valid_prefix
__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 937
__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 939
value
std::error_code, 2113
std::error_condition, 2115
std::regex_traits< _Ch_type >, 2829
value_comp
std::map< _Key, _Tp, _Compare, _Alloc >, 2405
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2499
std::multiset< _Key, _Compare, _Alloc >, 2523
std::set< _Key, _Compare, _Alloc >, 2952
value_compare

std::set< _Key, _Compare, _Alloc >, 2934
value_type
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1724
__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1729
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1739
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1742
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >, 2306
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >, 2310
std::allocator_traits< _Alloc >, 1034
std::allocator_traits< allocator< _Tp > >, 1040
std::allocator_traits< allocator< void > >, 1049
std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1057, 1058
std::back_insert_iterator< _Container >, 1077
std::complex< _Tp >, 1866
std::complex< double >, 1868
std::complex< float >, 1871
std::complex< long double >, 1874
std::front_insert_iterator< _Container >, 2162
std::insert_iterator< _Container >, 2253
std::istream_iterator< _Tp, _CharT, _Traits, _Dist >, 2292
std::istreambuf_iterator< _CharT, _Traits >, 2295
std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >, 2300
std::ostream_iterator< _Tp, _CharT, _Traits >, 2688
std::ostreambuf_iterator< _CharT, _Traits >, 2692
std::raw_storage_iterator< _OutputIterator, _Tp >, 2797
std::set< _Key, _Compare, _Alloc >, 2934
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3173
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3203
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3232
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3258
Variable template for type traits, 291
variant, 3787
vector, 3787–3789
std::__debug::vector< _Tp, _Allocator >, 3289
std::vector< _Tp, _Alloc >, 3293–3295
std::vector< bool, _Alloc >, 3313–3315
vector.tcc, 3536
version.h, 3536
void_pointer
__gnu_cxx::__alloc_traits< _Alloc, typename >, 735
std::allocator_traits< _Alloc >, 1034
std::allocator_traits< allocator< _Tp > >, 1040
std::allocator_traits< allocator< void > >, 1049
std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1058
void_t
Detection idiom, 285
vstring.h, 3675
vstring.tcc, 3678
vstring_fwd.h, 3679
vstring_util.h, 3680

wbuffer_convert
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3331, 3332
wcerr
std, 646
wcin
std, 646
wclog
std, 647
wcout
std, 647
wcregex_token_iterator
Regular Expressions, 255
wcsub_match
Regular Expressions, 255
weeks
Time, 366
wfilebuf
I/O, 166
wfstream
I/O, 166
what
__gnu_cxx::forced_error, 2135
__gnu_cxx::recursive_init_error, 2811
__gnu_pbds::container_error, 1897
__gnu_pbds::insert_error, 2251
__gnu_pbds::join_error, 2302
__gnu_pbds::resize_error, 2832
std::bad_alloc, 1079
std::bad_cast, 1081
std::bad_exception, 1082
std::bad_function_call, 1083
std::bad_typeid, 1084
std::bad_weak_ptr, 1085
std::domain_error, 2067
std::exception, 2117
std::experimental::filesystem::v1::filesystem_error, 2129
std::experimental::fundamentals_v1::bad_any_cast, 1080
std::experimental::fundamentals_v1::bad_optional_access, 1084

std::filesystem::filesystem_error, 2130
std::future_error, 2178
std::invalid_argument, 2256
std::length_error, 2312
std::logic_error, 2359
std::out_of_range, 2694
std::overflow_error, 2703
std::range_error, 2783
std::regex_error, 2817
std::runtime_error, 2844
std::system_error, 3046
std::underflow_error, 3142
widen
std::__ctype_abstract_base< _CharT >, 761
std::basic_fstream< _CharT, _Traits >, 1153
std::basic_ifstream< _CharT, _Traits >, 1196
std::basic_ios< _CharT, _Traits >, 1219
std::basic_iostream< _CharT, _Traits >, 1265
std::basic_istream< _CharT, _Traits >, 1303
std::basic_istream< _CharT, _Traits >::sentry, 2893
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1342
std::basic_ofstream< _CharT, _Traits >, 1376
std::basic_ostream< _CharT, _Traits >, 1406
std::basic_ostream< _CharT, _Traits >::sentry, 2920
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1440
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1710
std::ctype< _CharT >, 1922
std::ctype< char >, 1955–1957
std::ctype< wchar_t >, 1975, 1976
std::ctype_byname< _CharT >, 1989, 1990
std::ctype_byname< char >, 2011, 2012
width
std::basic_fstream< _CharT, _Traits >, 1154
std::basic_ifstream< _CharT, _Traits >, 1197
std::basic_ios< _CharT, _Traits >, 1219, 1220
std::basic_iostream< _CharT, _Traits >, 1265
std::basic_istream< _CharT, _Traits >, 1303, 1304
std::basic_istream< _CharT, _Traits >::sentry, 2893
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1343
std::basic_ofstream< _CharT, _Traits >, 1376, 1377
std::basic_ostream< _CharT, _Traits >, 1406, 1407
std::basic_ostream< _CharT, _Traits >::sentry, 2920
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1440
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1711
std::ios_base, 2265
wifstream
I/O, 166
wios
I/O, 166
wiostream
I/O, 166
wistream
I/O, 166
wistringstream
I/O, 166
wofstream
I/O, 166
workstealing.h, 3748
wostream
I/O, 166
wostringstream
I/O, 166
wregex
Regular Expressions, 255
write
std::basic_fstream< _CharT, _Traits >, 1154
std::basic_iostream< _CharT, _Traits >, 1266
std::basic_ofstream< _CharT, _Traits >, 1377
std::basic_ostream< _CharT, _Traits >, 1407
std::basic_ostream< _CharT, _Traits >::sentry, 2921
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1441
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1711
ws
std, 645
wsregex_token_iterator
Regular Expressions, 255
wssub_match
Regular Expressions, 255
wstreambuf
I/O, 167
wstreampos
std, 580
wstring
Strings, 273
wstring_convert
std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >, 3349, 3350
wstringbuf
I/O, 167
wstringstream
I/O, 167

xalloc
std::basic_fstream< _CharT, _Traits >, 1155
std::basic_ifstream< _CharT, _Traits >, 1197
std::basic_ios< _CharT, _Traits >, 1220
std::basic_iostream< _CharT, _Traits >, 1266
std::basic_istream< _CharT, _Traits >, 1304
std::basic_istream< _CharT, _Traits >::sentry, 2893

std::basic_istringstream< _CharT, _Traits, _Alloc >, 1343
std::basic_ofstream< _CharT, _Traits >, 1377
std::basic_ostream< _CharT, _Traits >, 1407
std::basic_ostream< _CharT, _Traits >::sentry, 2921
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1441
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1712
std::ios_base, 2265
xsgetn
__gnu_cxx::enc_filebuf< _CharT >, 2098
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 3006
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3024
std::basic_filebuf< _CharT, _Traits >, 1106
std::basic_streambuf< _CharT, _Traits >, 1471
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1666
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3343
xsputn
__gnu_cxx::enc_filebuf< _CharT >, 2099
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 3006
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 3025
std::basic_filebuf< _CharT, _Traits >, 1107
std::basic_streambuf< _CharT, _Traits >, 1472
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1667
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3343

years
Time, 366
yield
std::this_thread, 721